Departamento de Automática, Ingeniería Electrónica e Informática Industrial

Escuela Técnica Superior de Ingenieros Industriales

Estudio de la actividad de conmutación de circuitos electrónicos digitales descritos en el nivel de transferencia de registros mediante técnicas probabilísticas. Propuesta de un método de estimación

Autor: **Felipe Machado Sánchez**

Ingeniero Industrial por la Universidad Politécnica de Madrid

Director: **Yago Torroja Fungairiño**

Doctor Ingeniero Industrial por la Universidad Politécnica de Madrid

2008

Tribunal

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día 20 de junio de 2008.

Presidente:	Dr. Javier Uceda Antolín
Vocales:	Dr. Antonio Núñez Ordóñez
	Dr. Joan Figueras Pamiés
	Dr. Joao Paulo Teixeira
Secretario:	Dr. Carlos Alberto López Barrio
Suplentes:	Dra. Teresa Riesgo Alcaide
	Dr. Marcelino Bicho dos Santos

Realizado el acto de lectura y defensa de la Tesis Doctoral el día 17 de julio de 2008 en la Escuela Técnica Superior de Ingenieros Industriales de la Universidad Politécnica de Madrid.

Calificación:

EL PRESIDENTE LOS VOCALES

EL SECRETARIO

Reflexiones

Más de nueve años me ha llevado realizar La Tesis y más de la mitad de este tiempo he estado "terminando La Tesis". Ha sido un periodo duro, árido y con mucha soledad. En este tiempo he visto cómo he ido dejar pasar la Vida. He estado fines de semana, veranos y vacaciones "terminando La Tesis", sin permitirme disfrutar de La Vida, la familia, los amigos, parejas ... y de mí. He llegado a la desesperación y al enfado, y hasta le he "gritado" a Dios (si es que existe), sabiendo que, sin embargo, las llaves de los grilletes las tenía yo, y que en cualquier momento podía haber dejado la cárcel en la que inconscientemente me había metido. Esto me ha desesperado aún más, pues sabía que no había culpables externos. Sin saber por qué, nunca decidí dejar La Tesis. No sé si por terquedad, por orgullo o hasta quizás por miedo. Lo cierto es que al final nunca me quise liberar sin haber acabado La Tesis.

No considero que desde el punto de vista profesional o académico La Tesis haya valido lo que he pagado por ella. Sin embargo, La Tesis me ha enseñado mucho. Creo que hay dos vías para aprender: la vía del juego y la diversión, y la vía del dolor y el trabajo duro. Aunque normalmente en el proceso de aprendizaje no se toma una vía en exclusiva, desafortunadamente se puede decir que sólo escogí esta última vía. A pesar de todo, considero un tesoro todo lo que he aprendido y por esto no me lamento. Incluso, en la profundidad de mí, agradezco. Espero, eso sí, no olvidar y no tener que aprender más de la misma manera y ojalá que a partir de ahora decida y sepa aprender por la vía del juego, la diversión y el amor.

Agradecimientos

Han sido muchos años de tesis y en todo este tiempo he tenido el apoyo, el ánimo, la compañía y la comprensión de muchísima gente. Algunos me han acompañado durante todo el proceso, otros durante una parte. Este no sólo es un agradecimiento, sino un bonito recuerdo que compensa la dureza de estos años.

En primer lugar agradezco a mis padres su apoyo incondicional. Ellos han estado alentándome todo este tiempo y han sido capaces de aceptar y aguantar mis estados internos. Y cómo no, han sufrido la tesis, ya que muy a mi pesar, y sobre todo al suyo, pocas veces al año he podido ir por casa y dedicarnos el tiempo a estar juntos. Gracias.

En el aspecto tanto humano como técnico, quiero agradecer a Teresa Riesgo por sus constantes ánimos, su interés y su receptividad. Aunque quizá no se lo haya podido hacer ver, para mí ha sido muy importante. Gracias.

A Eduardo de la Torre por su disposición a la ayuda. En total, han sido muchas las horas que ha estado manteniendo las estaciones de trabajo, instalando las licencias de Synopsys,.... Sin esta ayuda, gran parte de mi trabajo se hubiese perdido. Gracias.

En este sentido, también le agradezco a Mario García cuando estuvo en el departamento, que igualmente me ayudó con los problemas relativos a las estaciones de trabajo siempre que se lo solicité. Igual que Edu, recuerdo que siempre lo hizo con muy buena disposición. Gracias.

A Fernando Casado, por la compañía y el trabajo en equipo que llevamos durante los primeros años de la tesis. Gracias.

En todos estos años he tenido muchos compañeros de doctorado y del departamento, a ellos quería recordar y agradecer los buenos momentos que hemos tenido: Fernando, Susana, Cristina, Mario, Ángel, Miguel, Marina, Alberto, Almudena, Ana, Nacho, Luís Alejandro, Pablo, Andrés, Coni, Ahmed, Jorge, Jaime, Yana, Vlado, Rubén, Eduardo, Yaseer, Miroslav, Carmen, Leonardo, ... Con ellos he pasado muy buenos momentos en el departamento. Gracias.

También quiero agradecer a todos los profesores y personal del departamento, por su apoyo y su ayuda. Gracias.

A mis nuevos compañeros de trabajo les agradezco su comprensión y sobre todo, el dejarme dedicar tiempo a terminar la tesis: Juan Antonio, Susana, Norberto, Belén, Bea, Ángel Luís, Joaquín, Cristina, Roberto, Juanfran, Araceli, Carmen, Miguel. Especialmente a aquellos que han tenido que trabajar más porque con la tesis no he podido involucrarme en proyectos y tareas organizativas. A "mi jefe" Juan Antonio por permitirme este tiempo y a Susana por animarme a encontrar un trabajo que me encanta. Gracias.

A mis compañeros del FZI de Karlsruhe, donde pasé un tiempo estupendo. A Andreas, Nati, Ralf, Felipe, Diana, el profesor Rosenstiel, ... Desde un principio me sentí muy acogido y como en casa. Gracias.

A la Dra. Anne-Marie Trullemans-Anckaert y al Dr. Vassilis Paliouras por acceder a participar como revisores internacionales de esta tesis. Gracias.

Por último, quiero recordar y agradecer a mis hermanos, amigos, ex-parejas, compañeros de pisos, ... que tanto me han acompañado en este tiempo, y que han sabido aguantar mi tensión, falta de tiempo, cansancio,... . A Todos, Muchas Muchas Gracias.

Índice

Lista de acrónimos

*a*BDD	*Activity BDD*
ALU	*Arithmetic Logic Unit*
ASIC	*Application Specific Integrated Circuit*
BDD	*Binary Decision Diagram*
BN	*Bayesian Network*
CAD	*Computer Aided Design*
CBDD	*Connective BDD*
CD	*Combinational Depth*
CEI	*Centro de Electrónica Industrial*
CMOS	*Complementary Metal-Oxide-Silicon (o Semiconductor)*
*da*BDD	*Distinct aBDD - aBDD diferenciado*
DBN	*Dynamic Bayesian Network*
DFT	*Design for Testability*
EDA	*Electronic Design Automation*
EHM	*Extended Hardware Model*
ESTG	*Extended State Transition Graph*
FA	*Full Adder*
HA	*Half Adder*
*ia*BDD	*Indistinct aBDD - aBDD Indiferenciado*
IP	*Intellectual Property*
ITE	*If-Then-Else*
ITRS	*International Technology Roadmap for Semiconductors*
MOS	*Metal-Oxide-Semiconductor (o Silicon)*
NMOS	*N-channel MOS*
OBDD	*Ordered Binary Decision Diagram*
PMOS	*P-channel MOS*
ROBDD	*Reduced Ordered Binary Decision Diagram*
RT	*Register Transfer*
RTL	*Register Transfer Level*
SAIF	*Switching Activity Interchange Format*
SHM	*Simplified Hardware Model*
SOP	*Sum-Of-Products*
STG	*State Transition Graph*
TBF	*Timed Boolean Function*
TFBDD	*Transition Function Binary Decision Diagram*
TDM	*Temporal Dependency Model*
TLM	*Transaction Level Modeling*

UPM	*Universidad Politécnica de Madrid*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*
VIF	*VHDL Intermediate Format*
VLSI	*Very Large Scale Integration*
ZBDD	*Zero suppressed BDD*

Lista de nombres de variables

Lista de figuras

Lista de gráficas

Lista de tablas

PLANTEAMIENTO Y RESUMEN DE LA TESIS

Durante las últimas décadas, la progresión de la electrónica digital ha sido extraordinaria. El continuo escalado de las tecnologías ha propiciado sorprendentes incrementos en la frecuencia de reloj y en las densidades de integración de los circuitos integrados. Esto ha implicado unos aumentos espectaculares de la funcionalidad y de las prestaciones de los circuitos. Sin embargo, la miniaturización de las tecnologías ha enfrentado a los diseñadores de circuitos con nuevos problemas, entre los que figura el consumo energético. Como consecuencia, los diseñadores no sólo se enfrentan con el diseño de circuitos de extraordinaria complejidad, sino que además deben considerar una serie de aspectos no relacionados con la funcionalidad, pero que son igualmente necesarios para la viabilidad del producto.

A estas dificultades propias del diseño se suman las exigencias del mercado, en donde la vida útil de los productos se acorta y los retrasos en la salida al mercado de un producto pueden ocasionar graves pérdidas económicas y de imagen.

En este contexto de dificultad, se hace patente la **necesidad de metodologías y herramientas de ayuda al diseño**. Metodologías y herramientas que faciliten un flujo de diseño continuo, de modo que no haya vueltas a atrás (iteraciones) causadas por el incumplimiento de alguna especificación. En muchas ocasiones, aunque el circuito sea funcionalmente correcto, estas iteraciones deben llevarse a cabo debido a que el circuito no cumple las especificaciones de aspectos importantes, como lo serían el consumo o las prestaciones.

Dentro de las herramientas de ayuda al diseño, los **estimadores constituyen un grupo fundamental**. Mediante la estimación se pueden anticipar las características finales del circuito, proporcionando al diseñador una información que de otro modo adquiriría en una fase más avanzada del diseño. Esto permite evitar costosas iteraciones en el flujo del diseño, ya que con la información facilitada por el estimador se pueden tomar las decisiones de manera anticipada, evitando continuar por un camino inútil por el que no se cumplirían las especificaciones.

Uno de los **principales** aspectos de un circuito que se debe de considerar durante el diseño es el **consumo energético**. Las enormes densidades de integración unidas a las altas frecuencias de reloj producen un altísimo consumo energético por unidad de superficie. Esto acarrea problemas térmicos que amenazan la integridad del dispositivo y hacen necesario el uso de elementos de refrigeración, lo que además disminuye la fiabilidad, aumenta el coste, el tamaño, el peso y el ruido del sistema.

Por otro lado, debido a la gran difusión de equipos móviles, se requieren dispositivos de bajo consumo para aumentar la autonomía y reducir el tamaño y peso de las baterías. El consumo energético es también un parámetro importante para reducir los costes de electricidad y sus consecuencias medioambientales.

Dentro del consumo total de un circuito digital, una parte importante es debida a la actividad de conmutación de los transistores y es aquí donde se enfoca la investigación de esta tesis doctoral: en **el estudio de la actividad de conmutación de los diseños digitales**. La actividad de conmutación de un circuito no sólo es un parámetro fundamental en el consumo, sino que además puede contribuir al análisis de otras características del circuito, como lo son la testabilidad y la fiabilidad.

El estudio de la actividad de conmutación no resulta fácil ya que no sólo depende del propio circuito, sino que también depende de la secuencia de los valores de las entradas. Para afrontar su análisis existen dos tipos de métodos: **dinámicos** y **estáticos**. Los métodos dinámicos son más sencillos de llevar a cabo ya que fundamentalmente se limitan a extraer las estadísticas de actividad de las señales a partir de simulaciones. El mayor inconveniente de estos métodos son los largos tiempos de simulación requeridos, que pueden resultar inviables para circuitos grandes. Es por ello que se han elaborado propuestas para reducir estos tiempos.

Los métodos estáticos, en su mayoría, elaboran modelos probabilísticos para el cálculo de las actividades de las señales. El mayor inconveniente de estos métodos es que para lograr un bajo

error en la estimación, la complejidad de los modelos resultantes es elevada. Para circuitos grandes se debe simplificar el modelo para que sea viable.

En esta tesis doctoral se ha optado por el método estático. Las razones que han llevado a esta elección son:

- La tendencia por los métodos estáticos para evitar los largos tiempos de simulación.
- El interés por continuar con la línea de investigación del departamento, aprovechando los recursos y conocimientos generados, y aplicar el método propuesto en otros análisis que contribuyan en la ayuda al diseño electrónico.

Las propuestas de análisis probabilístico de la actividad de conmutación han tenido como objetivo circuitos descritos en el nivel de puertas. La **aportación fundamental de esta tesis es subir el análisis desde el nivel de puertas al nivel de transferencia de registros (RTL)**. Con esto se consiguen varios objetivos:

- Poder realizar la estimación con anterioridad, lo que ayuda a evitar las iteraciones del diseño.
- Reducir la complejidad del análisis debido a que en el nivel RT la información no es tan exhaustiva.
- Aprovechar la información de más alto nivel que no está disponible en el nivel de puertas. Esta información permite simplificar el modelo probabilístico.

Subir el análisis al nivel de abstracción a RTL implica la aparición de nuevas dificultades con respecto al nivel de puertas, así como la oportunidad de aprovechar las ventajas que este nivel presenta. Por otro lado, también existen problemas comunes en ambos niveles, sobre todo teniendo en cuenta que en el nivel RT caben descripciones en puertas. Por tanto, el modelo propuesto no es una sustitución del modelo en puertas, sino que lo abarca y lo amplía.

Debido a estas dificultades y nuevas oportunidades, en esta tesis se formulan **una serie de propuestas particulares**:

- Como en el nivel RT no existe un modelo de dependencias e influencias tan explícito como en el nivel de puertas, en esta tesis **se propone la creación de un modelo** en el que se definan las dependencias e influencias de las señales con relación al hardware que se generará, y sobre el que se pueda llevar a cabo los análisis probabilísticos y realizar la propagación de las actividades.
- En la propuesta de esta tesis se emplean los *diagramas de decisión binaria* (BDD) para el cálculo de las actividades. El tamaño de los BDD depende del orden interno de sus variables. En esta tesis **se propone un ordenamiento de los BDD relacionado con la disposición de las señales en la descripción RTL, los operadores implicados y con los índices de las señales agrupadas en vectores**. Con este ordenamiento se consigue un **tamaño reducido de los BDD** que no es posible obtener de manera sencilla desde el nivel de puertas.
- Debido al enorme crecimiento de los BDD con el tamaño y la complejidad de los circuitos, éstos deben de dividirse (*particionarse*) para poder afrontar el análisis de manera eficiente. Sin embargo, no en todos los circuitos es posible realizar una partición que no conlleve error en el cálculo, por tanto el análisis probabilístico puede resultar inviable. En consecuencia, para circuitos grandes en los que no se pueda realizar esta partición, en esta tesis **se propone un método sencillo de dividir el circuito con el que si se cumplen unas serie condiciones planteadas en esta tesis, se logra un error bajo**.
- En esta tesis **se propone una nueva manera más compacta de representar los BDD de actividad**. Dentro de la propuesta se incluyen las operaciones de creación, transformación y manipulación de estos BDD.

Estas propuestas se han integrado en una herramienta informática. Esto ha permitido comprobar de manera práctica su viabilidad y además ha permitido la realización de

experimentos con circuitos medianamente grandes que, sin la existencia de esta herramienta, hubiesen sido imposible de llevar a cabo.

La memoria de esta tesis se ha dividido en cinco capítulos. En el primero se expone la importancia de la estimación de la actividad de conmutación. En el segundo capítulo se explican los distintos enfoques para el análisis de la actividad, se definen la terminología y fundamentos matemáticos del método probabilístico, y se detallan las propuestas más significativas dentro del análisis probabilístico. En el tercer capítulo se expone la propuesta de esta tesis doctoral. En el cuarto capítulo se muestran los resultados experimentales. Y para terminar, en el capítulo cinco se elaboran las conclusiones y líneas futuras de esta tesis.

En el **capítulo primero** se plantea la problemática del diseño de circuitos digitales y se evidencia la necesidad de metodologías y herramientas de ayuda al diseño. En este planteamiento se hace patente el importante papel que juegan los estimadores dentro de metodologías y herramientas de diseño. Además, en este capítulo se explica la importancia del consumo energético de los circuitos digitales y se exponen sus causas. A partir de estas causas se extrae la importancia de la actividad de conmutación y la necesidad de su estimación. En el capítulo también se presentan otras utilidades de la estimación de la actividad.

En el **capitulo segundo** se exponen los distintos métodos para la estimación de la actividad de conmutación y se fundamenta la elección del método probabilístico. Debido a la terminología específica del método probabilístico, en este capítulo se define esta terminología y se exponen las bases del método. Por último, se analiza el trabajo previo en este campo, situando dentro de este contexto a la propuesta formulada en esta tesis.

En el **capítulo tercero** se presenta el método propuesto en esta tesis para estimar la actividad de conmutación en el nivel RT. En este capítulo se propone el modelo probabilístico y el conjunto de propuestas particulares que sacan partido de realizar el estudio desde RTL, como son la partición disjunta y el ordenamiento RTL. En este capítulo también se propone una manera más eficiente de representar los BDD de actividad, estableciéndose las operaciones de creación, transformación y manipulación de estos nuevos BDD.

En el **capítulo cuarto** se muestran los resultados experimentales. Con la herramienta automática que se ha desarrollado se han podido analizar circuitos de mediana complejidad. En este capítulo se describe el método seguido para realizar las pruebas y los resultados obtenidos. Debido a la diversidad de propuestas de esta tesis y a que estas propuestas se pueden aplicar de manera independiente, la exposición de los resultados no es única, habiendo por tanto varios apartados que tratan estas propuestas individualmente y otros que exponen los resultados de manera conjunta.

En el **capítulo quinto** se elaboran las conclusiones generales de esta tesis doctoral y se sugieren las líneas futuras.

Además, esta tesis contiene **cuatro anexos**. El primero incluye información adicional sobre los diagramas de decisión binaria (BDD), que son ampliamente utilizados en esta tesis.

El **anexo segundo** contiene información adicional a la propuesta de esta tesis (capítulo tercero). La información contenida en este anexo no es necesaria para el entendimiento de la propuesta, pero plantea algunos aspectos del método que no se han tratado en el capítulo tercero para evitar su sobrecarga. Además, en este anexo se detallan algunos de los métodos y algoritmos empleados en la implementación de la propuesta. Esto puede ser de utilidad para aquellos que deseen implementar el método propuesto. Por último, en este anexo también se amplían algunos casos teóricos del análisis del error producido por la partición en regiones disjuntas.

En el **anexo tercero** se amplía la información relativa al modelo generado de los circuitos de pruebas y el análisis del error en estos circuitos debido a las particiones disjuntas propuestas en esta tesis.

En el **anexo cuarto** se muestra brevemente la implementación del método propuesto en una herramienta informática.

Por último, después de los anexos se incluye un extensa traducción de la tesis al inglés. Los dos primeros capítulos se han resumido en un capítulo llamado "E1: *Previous Work*". La numeración de la traducción ingles se ha precedido por la letra "E" para distinguirla de los capítulos en español. El resto de capítulos de la tesis se han traducido por completo, sin embargo, los anexos no se han traducido.

1. INTRODUCCIÓN

Como ya se ha comentado en el resumen de esta tesis, en este capítulo se introduce la problemática del diseño de los circuitos electrónicos digitales. Hoy en día, el diseño de circuitos digitales entraña tal complejidad que resulta imprescindible el uso de metodologías y herramientas de ayuda al diseño. Esta complejidad no sólo es debida a la enorme funcionalidad de los circuitos actuales, sino también al elevado número de aspectos que deben ser considerados en el diseño. Entre éstos, se encuentran aspectos como las prestaciones, el consumo, el área, la testabilidad, la fiabilidad,... En consecuencia, las metodologías y herramientas de ayuda diseño no sólo deben tratar la funcionalidad del circuito, sino todos los elementos que contribuyan a la realización exitosa del circuito.

Sin embargo, durante el diseño no resulta fácil anticipar todas las características finales del circuito. El incumplimiento de alguna de estas características puede hacer que, a pesar de que su funcionalidad sea correcta, el circuito no se considere aceptable. Por tanto, la existencia de estimadores que anticipen ciertas características del circuito es imprescindible para obtener el circuito en un tiempo adecuado. Cuanto antes se pueda realizar la estimación, menores retrasos en el diseño se producirán.

Actualmente, una de las características fundamentales de un circuito es su consumo energético, tanto para aumentar la autonomía de los dispositivos portátiles como para reducir la elevada disipación de calor que compromete la fiabilidad del circuito y aumenta los costes debidos a la refrigeración. Sin olvidar tampoco la importancia del consumo energético por los costes de electricidad y las consecuencias medioambientales.

En un circuito digital, la parte del consumo debida a su actividad se denomina consumo dinámico. Éste constituye una parte importante del consumo total de un circuito, y se produce por la actividad de conmutación de sus transistores. El objetivo fundamental de esta tesis es el estudio de la actividad de conmutación. No obstante, las implicaciones del estudio de la actividad de conmutación no están restringidas únicamente al consumo. Otros análisis pueden verse favorecidos por este estudio, como lo serían los análisis de testabilidad, área y de calidad.

En este capítulo se expone el contexto en el que se enmarca esta tesis. El primer apartado explica la problemática actual del diseño de los circuitos electrónicos digitales, en donde también se incluyen los principales retos del diseño. En el segundo apartado se expone la necesidad de estimadores dentro de las metodologías de diseño. A continuación, en el apartado 1.3 se explica la importancia y las causas del consumo. Es en este apartado donde se pone de manifiesto la trascendencia de la actividad de conmutación en el consumo. En el apartado 1.4 se exponen otros beneficios que el estudio de la actividad de conmutación puede aportar al diseño de circuitos digitales. Por último, en el apartado 1.5 se expondrán las conclusiones del capítulo.

1.1 El diseño de circuitos electrónicos digitales

Durante más de tres décadas, los circuitos electrónicos han experimentado una vertiginosa evolución. La continua miniaturización de las tecnologías ha permitido un aumento exponencial de las capacidades de los circuitos, produciendo espectaculares incrementos en las frecuencias y densidades de integración. Como consecuencia de este desarrollo, los circuitos integrados actuales son extremadamente complejos, tanto por el número de transistores y sus interconexiones, como por la funcionalidad de los bloques que contienen. Por esto, resultan imprescindibles metodologías y herramientas adecuadas que asistan en el manejo de tal complejidad.

Sin embargo, la realidad es que la capacidad de diseño no ha avanzado acorde con la capacidad tecnológica. Y aunque ha habido un incremento muy significativo en la productividad de

diseño[1], existe un desfase de productividad entre las capacidades tecnológicas y las capacidades de diseño [63], [43], [47], [54], [97], [114], [115].

Este desfase hace que el coste de diseño constituya la mayor amenaza para que se cumplan las expectativas de progreso en la industria microelectrónica (ITRS 2007 [63]). Los costes determinan en qué tecnología se implantará el diseño: si es mejor implementarlo en software o hardware; en una plataforma programable o en un circuito integrado. Mientras los costes de fabricación de ingeniería no-recurrente son del orden de millones de dólares (máscara y placa de prototipo); los costes de diseño alcanzan las decenas de millones de dólares, al incluirse las deficiencias del diseño que son responsables de tener que volver a enviar el diseño a fabricación.

Además, debido a los rápidos cambios tecnológicos, la vida de los productos se acorta y hace que el tiempo de salida al mercado sea un parámetro crítico. En este sentido, los tiempos de fabricación se miden en semanas, con muy poca incertidumbre. Mientras que los tiempos de diseño y verificación se miden en meses o años, con una gran incertidumbre [63].

Los retrasos en el diseño son muy costosos. En un mercado de rápido movimiento, un retraso de seis meses en un producto podría suponer casi un 50% de pérdida de ingresos, mientras que las pérdidas por un retraso de un año superarían el 90%. Estas pérdidas no sólo están causadas por la oportunidad de mercado perdida durante el retraso, en el que los competidores ganan cuota de mercado en el periodo más rentable, sino que además, una vez que sale el producto, éste debe ser vendido a menor precio por su llegada tardía al mercado. A esto hay que añadir los costes adicionales debidos a que los recursos de ingeniería de la empresa están dedicados a este proyecto durante un tiempo más prolongado del estimado [65].

Con el resultado de costes de desarrollo más elevados e ingresos menores, las consecuencias son grandes pérdidas y la incapacidad para competir en el mercado. Con tan solo dos o tres fabricantes de circuitos integrados capaces de generar beneficios financieros en cualquier segmento de mercado, llegar tarde al mercado suele resultar desastroso [65].

Ejemplos de datos extraídos de la bibliografía ([61], [62], [63], [97]) reflejan la situación actual del diseño electrónico: los costes de test del sistema han crecido exponencialmente respecto a los de fabricación; hasta un 85% de los proyectos de diseño tienen retrasos, con un retraso de 53% del tiempo originalmente estimado; de media hay más de cuatro rediseños (iteraciones) para completar el diseño, etc.

Estos datos convierten a las herramientas y metodologías de ayuda al diseño en una prioridad para el desarrollo de la microelectrónica, haciendo que los resultados financieros de las compañías de semiconductores estén directamente relacionados con sus capacidades de diseño [4], [63], [97].

En el siguiente subapartado se expondrán las principales áreas donde se deben enfocar las mejoras en la tecnología de diseño.

1.1.1 Principales retos en el diseño

Existen dos tipos de dificultades que el diseño de circuitos integrados tiene que enfrentar, la complejidad del silicio y la complejidad del sistema.

Con el término **complejidad del silicio** se hace referencia al impacto de la miniaturización de las tecnologías y de la introducción de nuevos materiales y dispositivos.

Existen muchos fenómenos que antes eran insignificantes y que ahora tienen un gran impacto en el diseño debido a que, entre otras causas, los elementos parásitos del transistor y las tensiones alimentación/umbral no se escalan idealmente. Por ejemplo actualmente los

[1] Productividad de diseño: número de puertas lógicas o transistores que un diseñador es capaz de diseñar por día

acoplamientos entre interconexiones y transistores ya no son despreciables. A esto se añade que existe una gran *variabilidad* en la fabricación, que ocurre tanto entre dado y dado de silicio de una misma oblea, como entre dados de distintas obleas [63].

Estos fenómenos aumentan la complejidad y ponen en riesgo los hasta ahora sólidos fundamentos del diseño, ya que la sincronización total del sistema resulta inviable, el comportamiento del transistor es muy variable y la fabricación de circuitos con la totalidad de sus transistores e interconexiones funcionando correctamente es prohibitivamente cara [63].

Estos retos demandan diseñadores con amplios conocimientos, y metodologías y herramientas con mayor fusión entre áreas hasta ahora separadas en el diseño, como síntesis y análisis, o como diseño lógico y físico.

La **complejidad del sistema** se refiere a las dificultades del diseño debidas al incremento exponencial del número de transistores que la miniaturización de las tecnologías permite. Esta complejidad está estimulada por la demanda de mayor funcionalidad, menor coste y menores tiempos de puesta en mercado. Esta demanda de mayor funcionalidad tiene un elevado impacto en la complejidad del sistema, ya que como enuncia la *"ley" de funcionalidad observada* [30] para lograr un aumento lineal de utilidad de un dispositivo electrónico, el aumento tecnológico (número de transistores, capacidad computacional) debe ser exponencial.

Para afrontar las dificultades que la complejidad del sistema implica, se precisa realizar avances en [63]:

- *Reutilización* de componentes y bloques de propiedad intelectual (IP[2]).
- Metodologías y herramientas que consideren desde el principio la *verificación* y el *test*.
- *Optimización* concurrente de múltiples parámetros del diseño: consumo, retardos, verificabilidad, testabilidad, tolerancia a fallos,...
- Diseño de sistemas empotrados.
- Implementación fiable del diseño en diferentes plataformas.
- Mejorar la *gestión* del proceso de diseño.

Estas dos complejidades implican un **incremento super-exponencial de la complejidad del diseño**. La tecnología de diseño debe optimizar y analizar concurrentemente los objetivos y restricciones más complejas, considerando adicionalmente la reutilización y los costes de fabricación; y abarcando el diseño de software empotrado y los interfaces con la fabricación.

Englobando todas las dificultades enumeradas, tres desafíos generales se pueden extraer como los más críticos para la tecnología de diseño: **productividad, consumo y capacidad de fabricación** (*manufacturability*) [63].

El primero está causado por el gran espectro que cubren estas complejidades, que, por sí mismo, constituye un reto para la tecnología de diseño y para la industria de automatización de diseño electrónico (EDA[3]), y el que hace que la **productividad del diseño sea el problema más acuciante tanto a corto como a largo plazo**.

El *consumo* es un reto urgente a corto plazo que, siendo fundamental como consumo dinámico para los circuitos de altas prestaciones, rápidamente está tomando interés en el consumo estático debido a la miniaturización y la variabilidad de la fabricación.

La *capacidad de fabricación* es la capacidad de producir un circuito integrado en grandes cantidades a un coste aceptable y en un tiempo económicamente viable. Este reto apunta a constituir una gran crisis debido a la variabilidad que en múltiples formas afecta todos los aspectos del diseño.

[2] Del inglés: *Intellectual Property*
[3] Del inglés: *Electronic Design Automation*

Concluyendo esta sección, todos los elementos analizados recalcan la importancia y necesidad de un avance significativo en las herramientas de ayuda al diseño electrónico. Siendo particularmente importantes aquellas relacionadas con el consumo energético, que no sólo es un aspecto crítico en la actualidad, sino que cada vez va ganando mayor importancia.

1.2 La estimación en el diseño electrónico

En este entorno de presión y complejidad que se acaba de describir, se precisan técnicas, herramientas y una metodología eficaz con los que a partir del trabajo de diseño se llegue de manera previsible a un producto fabricable.

Una metodología de diseño es la secuencia de pasos con los que un proceso de diseño producirá de manera fiable un diseño *tan cercano como sea posible* al objetivo de diseño, a la vez que conserva viabilidad respecto a las restricciones. Las técnicas de diseño forman parte de los pasos que constituyen la metodología. Todas las metodologías de diseño combinan [63]:

- Imposición de las especificaciones del sistema y restricciones por medio de planificación y búsqueda descendente.
- Propagación ascendente de las restricciones que son consecuencia de las leyes físicas, de los límites del diseño y de la tecnología de fabricación, así como las restricciones que surgen del presupuesto.

Mientras que se presta bastante atención a las herramientas, el aspecto metodológico, siendo igualmente importante, ha sido descuidado. Para cada nueva generación tecnológica es preciso considerar más aspectos. Consecuentemente, nuevas herramientas, análisis y métodos deben ser desarrollados para evaluar los nuevos fenómenos y ayudar al diseñador a tomar las decisiones críticas. Y más crítico aún es la determinación de la secuencia más eficiente para el diseño y toma de decisiones, de manera que se minimicen las iteraciones del diseño.

Las **iteraciones** del diseño se producen al detectar el incumplimiento de las especificaciones al finalizar una etapa, siendo necesario por tanto revisar parcial o totalmente el modelo de las etapas superiores para lograr el cumplimiento de dichas especificaciones. Cuanto **antes** se descubran estos errores, **menos dramáticos serán los cambios a realizar**, con el consiguiente ahorro en tiempo y esfuerzo de diseño. La figura 1-1 muestra las etapas típicas de diseño de una metodología en las que la revisión del diseño al finalizar cada etapa puede llevar a la modificación del modelo en etapas superiores [130].

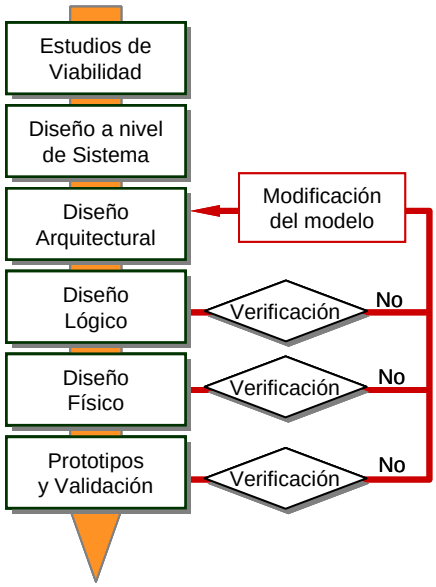


Figura 1-1: Bucles en el flujo de diseño

Por tanto es fundamental proporcionar **herramientas de estimación** desde las primeras etapas de diseño, para comprobar cuanto antes el cumplimiento de las especificaciones y proporcionar medios de comparación entre distintas opciones de diseño. En etapas donde el diseñador no tiene información suficiente, la estimación le permite la toma anticipada de decisiones, contribuyendo a lograr un **flujo de diseño continuo**, sin las iteraciones debidas a la detección de especificaciones incumplidas en etapas posteriores.

Además, ya se verá en el apartado 1.3.2 que cuanto más alto sea el nivel de abstracción, más influencia se tiene sobre el circuito. El disponer de la información proporcionada por el estimador en niveles altos de abstracción, permite optimizar el circuito con menor esfuerzo que desde niveles más bajos.

La estimación exige un compromiso entre **sencillez** y **precisión** [130]. Por un lado, debe estar basada en procedimientos sencillos de manera que aporten un beneficio en cuanto a recursos y tiempo de procesamiento de la medida real (si es que esta medida es posible). Y por otro lado, debe ser precisa, de modo que se acerquen lo más posible al valor real de la característica que se desea estimar.

1.3 El consumo en los circuitos electrónicos digitales

Desde hace unos años el consumo energético de los circuitos integrados digitales ha ido cobrando mayor importancia hasta convertirse en uno de los aspectos de mayor trascendencia durante su diseño [63], [110], [127], [116]. La reducción del consumo es necesaria tanto para los circuitos de equipos portátiles como para aquéllos de altas prestaciones.

Los equipos portátiles e inalámbricos requieren larga duración de sus baterías con tamaño y peso mínimos. Las necesidades cada vez mayores de computación y prestaciones de estos equipos precisan de circuitos con mínimo consumo, así como de la búsqueda de alternativas en los sistemas de alimentación [109].

Los equipos de altas prestaciones requieren minimizar el consumo para evitar una excesiva disipación de calor que provoca un aumento de temperatura que dañaría al sistema. Los sistemas de refrigeración y disipadores evacuan el calor evitando un aumento excesivo de temperatura, como contrapartida, el uso de sistemas de refrigeración compromete la fiabilidad del sistema, aumenta los costes y genera además otros inconvenientes como mayor tamaño y ruido. Aún así, las altas intensidades de cómputo actualmente demandadas pueden hacer necesaria la introducción de nuevos materiales con mayor conductividad térmica y técnicas de refrigeración más sofisticadas. Para ilustrar la magnitud de esta situación, se puede citar el caso del microprocesador Pentium 4, que consume más de 100 vatios, lo que dividido por su superficie, supone un flujo de unos 30 vatios por centímetro cuadrado, unas tres veces más que una placa de cocina eléctrica común [110], [75], [124].

Además, este incremento de temperatura, causado por tan alta disipación energética por unidad de superficie, hace que se reduzca la movilidad del electrón, y por tanto, que aumenten los retardos del circuito. El aumento de los retardos puede provocar que el circuito funcione de forma incorrecta por no cumplirse los tiempos de propagación de la señal.

Las altas temperaturas no sólo pueden provocar errores de funcionamiento debidos al aumento de los retardos, sino que también pueden conducir a la aparición de daños permanentes en el circuito. Por tanto, el elevado consumo afecta también a la fiabilidad y tiempo de vida del circuito. El estrés térmico a que se ven sometidos los circuitos integrados puede literalmente desgarrar el circuito, no tanto el silicio, sino el encapsulado. Por otro lado, las altas temperaturas de las interconexiones de cobre y aluminio hacen que sean más susceptibles a la desintegración en un fenómeno conocido como electromigración [19], lo que supone un serio inconveniente de cara a la fiabilidad del circuito.

La disipación energética puede estar más intensamente localizada en determinadas áreas del circuito, originando puntos calientes que lo hiciesen funcionar incorrectamente. Esto puede

suceder incluso cuando el consumo global del circuito no sea especialmente crítico, ya que se han detectado diferencias de hasta 30ºC entre distintas zonas de algunos microprocesadores [92].

Independientemente de los problemas de funcionamiento, fiabilidad, costes y los demás inconvenientes relativos al consumo que se han mencionado, en términos macro-energéticos el consumo también es una cuestión destacable. Especialmente para los grandes centros informáticos, en donde el elevado consumo es un problema debido a los costes de energía en alimentación y refrigeración. Así, un centro de cálculo con 1000 bastidores, con una superficie de más de 2000 metros cuadrados, requeriría unos 10MW de potencia para alimentar a los sistemas informáticos; además de otros 5MW para evacuar el calor disipado. A 100$/MWh, solamente el gasto de refrigeración supondría 4 millones de dólares anuales. Para mejorar la eficiencia energética de estos centros, se han propuesto complejos sistemas de refrigeración, administración de equipos y reparto de cargas de trabajo entre distintos centros de cálculo [110], [105], [88].

El interés por la eficiencia energética también se debe a las preocupaciones medioambientales surgidas por el aumento de las emisiones de gases contaminantes, el cambio climático y la escasez de recursos energéticos, entre otras. La gran difusión de los equipos electrónicos hace que, aunque localmente parezca inapreciable, a nivel global sea importante [88], e impele a la reducción de consumo en todo el rango de dispositivos y sistemas.

Recientemente, a raíz de la continua miniaturización de la tecnología CMOS digital, el problema se ha agravado aún más, haciendo incluso peligrar la tendencia que la microelectrónica ha experimentado durante más de tres décadas. El progreso exponencial de la microelectrónica se ha expresado con la "ley" de Moore, que en su enunciado más popular asevera que cada 18 meses se dobla el número de componentes en un circuito integrado [86].

Idealmente, tal como Dennard [33] expuso en 1974, si el voltaje se escalase junto con las dimensiones litográficas, se lograrían los beneficios que se asumen con la miniaturización: mayor velocidad, menor consumo y puertas más baratas. El incremento de gasto energético debido al aumento de puertas y su mayor frecuencia de conmutación se contrarrestaría con la disminución de energía empleada en cada conmutación. Así que, en teoría, el consumo por unidad de área permanecería constante.

Desafortunadamente, debido a que la miniaturización ha divergido de las relaciones ideales propuestas por Dennard, la densidad de consumo ha aumentado en vez de permanecer constante [56], [55]. Esto se debe a que la tensión umbral no se puede escalar de la misma manera que las dimensiones del transistor a causa de la aparición de corrientes de fugas [44], [67] y ruido térmico [68]. Como resultado, especialmente para microprocesadores de altas prestaciones, la tensión de alimentación se ha escalado en x0,85 en vez de x0,7 para cada generación tecnológica, mientras que la frecuencia se ha incrementado al doble en vez de a una tasa de x1,4 debido al uso intensivo de la segmentación (*pipeline*) y no enteramente por el aprovechamiento de las ventajas tecnológicas. El resultado es que se ha disminuido la cantidad de trabajo que se realiza por etapa de segmentación [63], [56], [55].

Como consecuencia, el consumo energético ha ido creciendo tanto por los incrementos en área como por los aumentos en densidad de consumo. El aumento de las pérdidas por las corrientes de fugas que se producen en cada generación tecnológica unidos a otros factores como variabilidad y ruido térmico, podrían provocar una crisis en la industria electrónica [63]. Esto ha generado un debate abierto sobre cuándo llegará esta crisis, sobre la continuidad de la tecnología CMOS y sus sucesores [87].

Una vez que se ha remarcado la importancia del consumo energético en los circuitos digitales CMOS, a continuación se detallarán sus causas y las estrategias para minimizarlo durante el diseño. Asimismo se establecerá la estrecha relación existente entre el consumo y la actividad de conmutación, cuyo análisis es la parte fundamental de esta tesis doctoral.

1.3.1 Causas del consumo

En este apartado se explicarán las causas de la disipación de energía en los circuitos integrados digitales CMOS estáticos[4], para así obtener los fundamentos de la estimación de consumo.

Las causas principales que provocan la disipación de energía en los circuitos digitales CMOS son:

- La carga y descarga de las capacidades internas, P_{dinc}([5])
- Las corrientes de corto-circuito, P_{sc}
- Las corrientes de fugas (*leakage*), P_{leak}

El consumo se puede agrupar en dinámico y estático. El dinámico está formado por el consumo debido a las corrientes de cortocircuito y por el consumo debido a la carga y descarga de las capacidades internas. Para acortar, a este último lo llamaremos *consumo dinámico capacitivo*.

En los siguientes apartados se detallarán las causas de estos tres tipos de consumo.

1.3.1.1 Carga y descarga de las capacidades internas

En la tecnología CMOS estática, cada vez que hay una transición de una señal se producen pérdidas energéticas debido a la carga o descarga de las capacidades internas de los transistores. Para mostrar este efecto, en vez de extraer un modelo preciso que tenga en cuenta todos los efectos internos del funcionamiento de los transistores, habitualmente se usa un modelo sencillo que consiste en concentrar todas las capacidades internas en los nodos lógicos del circuito [42], dando como resultado un modelo con un condensador de carga C_L a la salida de cada nodo. En términos del consumo dinámico capacitivo de la puerta, este modelo se comporta de manera similar al del circuito real.

Así, los circuitos equivalentes para un inversor son los mostrados en la figura 1-2. En ellos, si el transistor está dibujado con línea de puntos indica que el transistor está en corte. Cada vez que hay una transición de '0' a '1' (0→1) en la salida del inversor[6] (v_{out}), el condensador C_L se carga desde 0 a V_{dd} (ver figura 1-2-a). Para que esto tenga efecto, una cierta cantidad de energía se extrae de la fuente de alimentación. Parte de esta energía se disipa por el transistor PMOS y el resto se almacena en C_L.

Por otro lado, durante la transición 1→0 en la salida del inversor, la energía almacenada en el condensador C_L se disipa en el transistor NMOS (figura 1-2-b).

[4] La tecnología CMOS estática es la escogida para la mayoría de los diseños digitales actuales [106]. De todas maneras, para la tecnología CMOS dinámica muchos planteamientos son extensibles con ciertas correcciones

[5] Aunque tanto ésta como la siguiente son dinámicas, se ha dejado el subíndice *dinc* para seguir la nomenclatura más comúnmente utilizada. Para evitar confusión se ha añadido una *c* de "capacitiva"

[6] A partir de ahora, al hablar de transiciones en el inversor, si no se especifica lo contrario, se considerarán transiciones a la salida., i.e. una transición 0→1 será una transición 0→1 para v_{out}, mientras que para v_{in} será una transición 1→0

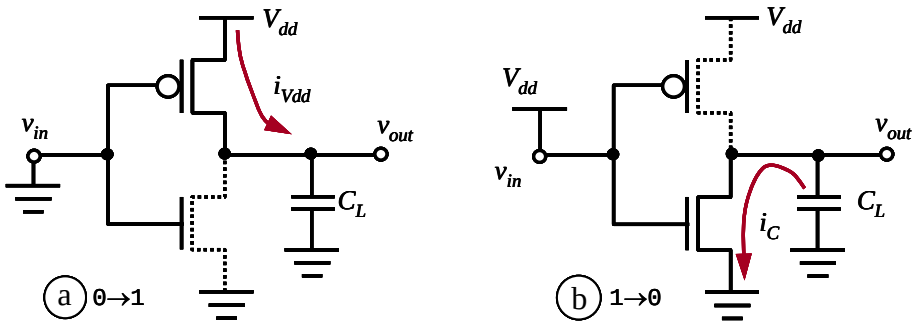


Figura 1-2: Carga y descarga del condensador a la salida del inversor debida a la conmutación

Para cada conmutación, ya sea 0→1 ó 1→0, se consume una cantidad fija de energía igual a ½ $C_L \cdot V_{dd}^2$ (véase el desarrollo en la referencia [42]). Para calcular la **potencia dinámica capacitiva disipada** por la puerta habrá que multiplicar la energía disipada en cada transición por la frecuencia de estas transiciones:

$$P_{dinc} = \tfrac{1}{2} \cdot C_L \cdot V_{dd}^2 \cdot \frac{n_t(T)}{T} \tag{1.1}$$

$n_t(T)$ representa el número de conmutaciones (0→1 ó 1→0) durante el intervalo de tiempo (-T/2, +T/2].

Llevando al límite la ecuación 1.1 se obtiene la potencia media:

$$P_{dinc} = \lim_{T\to\infty}\left[\tfrac{1}{2} \cdot C_L \cdot V_{dd}^2 \cdot \frac{n_t(T)}{T}\right] = \tfrac{1}{2} \cdot C_L \cdot V_{dd}^2 \cdot \lim_{T\to\infty}\left(\frac{n_t(T)}{T}\right) \tag{1.2}$$

El último término de la ecuación 1.2 es la media del número de transiciones por unidad de tiempo, Najm [90] lo define como densidad de transición, D:

$$D \equiv \lim_{T\to\infty}\frac{n_t(T)}{T} \tag{1.3}$$

Para circuitos síncronos[7] es habitual definir la frecuencia de conmutación en referencia a la frecuencia de reloj del circuito f_{clk}. En esta tesis se llamará **actividad de conmutación** (a) de una señal a la media de transiciones por ciclo de reloj ([42], [46]), y para circuitos asíncronos se supondrá la existencia de un ciclo mínimo del circuito.

$$a = D / f_{clk} \tag{1.4}$$

La ecuación de la potencia consumida por un inversor CMOS queda:

$$P_{dinc} = \tfrac{1}{2} \cdot a \cdot C_L \cdot V_{dd}^2 \cdot f_{clk} \tag{1.5}$$

De esta última ecuación (1.5) se pueden extraer todos los factores que influyen en el consumo de un inversor CMOS causado por la carga y descarga de las capacidades internas debidas a la actividad de conmutación.

Para el resto de puertas lógicas, la mayoría de los autores extienden el modelo anteriormente expuesto (por ejemplo [42], [90], [46], [22], [79]).

Una vez que se dispone de la información del consumo de la carga y descarga de las capacidades internas para una puerta se podrá extender el análisis para todo el circuito sumando los consumos individuales de cada puerta. Para el cálculo de la fórmula general bastará con realizar la sumatoria de las potencias consumidas por cada puerta i a partir de la ecuación 1.5

[7] En esta tesis se empleará el término *síncrono*, ampliamente usado en la literatura específica en español, en vez de *sincrónico*, que sería el término correcto según la Real Academia Española [108]

$$P_{dinc} = \tfrac{1}{2} \cdot V_{dd}^2 \cdot f_{clk} \cdot \sum_{i=1}^{n} (a_i \cdot C_{Li}) \quad (1.6)$$

En la ecuación 1.6, *n* representa el número de puertas del circuito. Con el avance tecnológico, cada vez se pueden integrar mayor número de transistores, superando los mil millones de transistores en un mismo dado de silicio. Asimismo, la frecuencia f_{clk} alcanza valores cada vez más altos conforme progresa la tecnología (superiores a 3 GHz). Por otro lado las capacidades internas (C_L) disminuyen a medida que se reduce la escala de los transistores. Esta reducción de escala hace que la tensión umbral disminuya, permitiendo bajar la tensión de alimentación V_{dd} ([106], [66]). Como se puede ver, esta reducción es muy importante por el efecto cuadrático que tiene sobre el consumo energético.

A modo de resumen se expondrán los factores de que depende la potencia dinámica debido a la carga y descarga de las capacidades internas:

- La tensión de la fuente de alimentación V_{dd}. Dependencia cuadrática, siendo por tanto un método muy efectivo para reducir este consumo.
- Para cada nodo es proporcional a su capacidad de carga C_L.
- Proporcional a la frecuencia de reloj f_{clk}.
- Para cada nodo es **proporcional a su actividad de conmutación** *a*.
- Dependiente del número de nodos (tamaño del circuito)

1.3.1.2 Corrientes de cortocircuito

Para los circuitos reales, la suposición de que las transiciones a la entrada de los transistores son instantáneas no es exacta. La pendiente finita de la señal de entrada provoca corrientes de cortocircuito[8] entre la fuente de alimentación V_{dd} y tierra V_{GND}. Durante un corto periodo de tiempo, los transistores PMOS y NMOS conducen simultáneamente al conmutar alguna señal de entrada. En la figura 1-3 se muestra este efecto particularizado para un inversor, aunque los resultados se extienden a otras puertas de manera cualitativa.

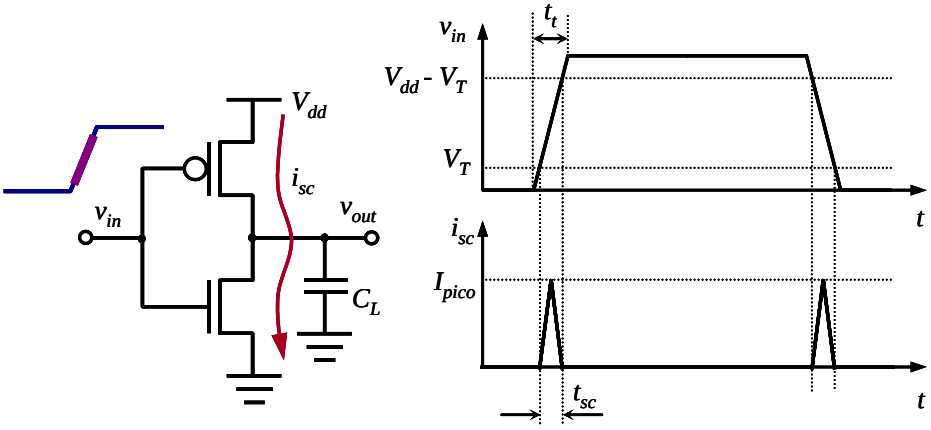


Figura 1-3: Corrientes de cortocircuito durante la conmutación

En la figura 1-3 la gráfica superior se corresponde con la tensión de entrada del inversor v_{in} respecto al tiempo, y la inferior se corresponde con la corriente de cortocircuito i_{sc} respecto al mismo tiempo que el de la gráfica superior.

Se llama t_t al tiempo total de la transición; t_{sc} representa el tiempo en el que ambos transistores están conduciendo ($t_{sc} < t_t$); mientras que V_T es la tensión umbral (*threshold*), que para simplificar se ha supuesto la misma para los transistores PMOS y NMOS ($V_{Tp} = V_{Tn}$).

[8] También llamadas corrientes de camino directo (*direct-path*)

Asumiendo que los picos de corriente se aproximan a triángulos y que el comportamiento del inversor es simétrico para las transiciones de subida y de bajada, la energía consumida en cada conmutación por el inversor resulta [106]:

$$E_{sc} = \int i_{sc} \cdot V_{dd} \cdot dt = \tfrac{1}{2} \cdot I_{peak} \cdot V_{dd} \cdot t_{sc} \tag{1.7}$$

Suponiendo que la transición de la señal de entrada es una recta de pendiente constante, t_{sc} resulta:

$$t_{sc} = \frac{V_{dd} - 2 \cdot V_T}{V_{dd}} \cdot t_t \tag{1.8}$$

Para el cálculo de su potencia media:

$$P_{sc} = \tfrac{1}{2} \cdot a \cdot I_{pico} \cdot V_{dd} \cdot t_{sc} \cdot f_{clk} \tag{1.9}$$

De la misma manera que en la disipación por la carga y descarga de las capacidades parásitas el consumo debido a la corriente de cortocircuito es también proporcional a la actividad de conmutación.

La corriente de pico I_{pico} depende de:

- La corriente de saturación de los transistores, siendo por tanto directamente proporcional a su tamaño (proporcional a β)
- La capacidad de carga C_L (I_{pico} decrece cuando C_L aumenta)

Según la ecuación 1.8, a medida que la tensión de alimentación disminuye, también lo hace t_{sc}, y por tanto disminuyen las pérdidas energéticas por las corrientes de cortocircuito (ecuación 1.7), y en el caso extremo cuando $V_{dd} < V_{Tn} + |V_{Tp}|$ la corriente de cortocircuito se elimina completamente porque los transistores nunca conducen simultáneamente. Conforme disminuye la escala de los transistores, disminuyen tanto la tensión de alimentación V_{dd} como la tensión umbral V_T, manteniéndose constante la relación V_T / V_{dd} ; ó incluso subiendo ligeramente para mantener el consumo estático en niveles tolerables (apartado 1.3.1.3) [99], [106]. Debido a esto, conforme se miniaturizan las dimensiones de los transistores, el consumo por corrientes de cortocircuito es menos significativo, o a lo sumo, se mantiene respecto al consumo dinámico total. La contribución del consumo por corrientes de cortocircuito al consumo dinámico total ($P_{dinc} + P_{sc}$) se mantiene alrededor de un diez por ciento.

1.3.1.3 Corrientes de fugas

El consumo debido a las corrientes de fugas se produce en condiciones estáticas, cuando todas las entradas y valores internos mantienen valores de tensión constantes. Las corrientes de fugas quiescentes I_{leak} fluyen desde la fuente a tierra provocando pérdidas energéticas conocidas como *consumo estático*, que en una expresión muy simplificada se formula [106]:

$$P_{leak} = I_{leak} \cdot V_{dd} \tag{1.10}$$

Idealmente, las corrientes de fugas en los dispositivos CMOS son nulas debido a que los transistores PMOS y NMOS no conducen simultáneamente en condiciones estáticas (sin transiciones). Esta suposición se podía considerar correcta hasta que, conforme ha avanzado la tecnología, la miniaturización de la escala de los dispositivos CMOS ha hecho aumentar la influencia de estas corrientes de fugas en el consumo total. Así, a diferencia de los consumos anteriormente expuestos, el consumo estático de un circuito no depende de la conmutación de sus transistores sino del número de transistores del circuito.

La fuerte reducción de la escala de los transistores conlleva efectos no deseados debidos a que no todas las características se escalan de forma ideal. La tensión umbral V_T no puede ser escalada en la misma proporción debido a que con ella aumentan exponencialmente las **corrientes de fuga sub-umbral** (*sub-threshold leakage current*). Por otro lado, ya que la tensión umbral no es proporcionalmente escalada con la medida del transistor, los retardos tampoco

disminuyen en la relación ideal. Consecuentemente, el efecto de la miniaturización exige un compromiso entre consumo estático y retardos [1], [111].

Desgraciadamente, la excesiva miniaturización comporta otros efectos indeseados, que se traducen en una mayor impredecibilidad en el comportamiento de los transistores. Para evitarla se toman dos medidas: disminuir el grosor del óxido de la puerta y aumentar el dopado[9]. La primera aumenta las **corrientes de fuga de la puerta** (*gate oxide tunnelling leakage*) y la segunda las **corrientes de fuga de banda a banda** (*band-to-band tunnelling*).

Aunque son las más importantes, las pérdidas por fugas no se reducen a las tres anteriormente citadas. Con la miniaturización de la tecnología, la aportación de las distintas pérdidas por fugas empieza a ser parecida, lo que, añadido a la dependencia de estas pérdidas de la geometría, contaminación y temperatura, hace que el análisis de las pérdidas por fugas se complique.

La corriente de fuga sub-umbral es la más dependiente de la temperatura [1]. Además, debido a que la corriente de fuga sub-umbral depende de las tensiones internas del transistor, ésta se hace **dependiente de los valores lógicos de la entradas** de las puertas [39], [25].

1.3.2 Optimización del consumo

En el diseño de circuitos integrados es preciso seguir una metodología adecuada que desde las primeras etapas persiga no sólo un funcionamiento correcto, sino el cumplimiento de las restricciones en cuanto consumo, prestaciones, área, costes, etc.

Durante estas etapas iniciales es cuando más incidencia sobre el consumo y otras características del circuito se puede ejercer. Mientras que conforme avanza el diseño, menor capacidad se tiene para influir en el consumo [11], [70], [56]. La figura 1-4 ilustra la capacidad de influir en el consumo de las diferentes etapas. Como puede apreciarse, la implementación en diseños y arquitecturas completamente distintas puede hacer el circuito de diez a veinte veces más eficiente en consumo (o incluso más [70]). Por otro lado, optimizaciones en las últimas etapas pueden lograr reducciones adicionales de un 10-20%. Esto es cualitativamente aplicable a otras características del diseño como prestaciones, área, costes,...

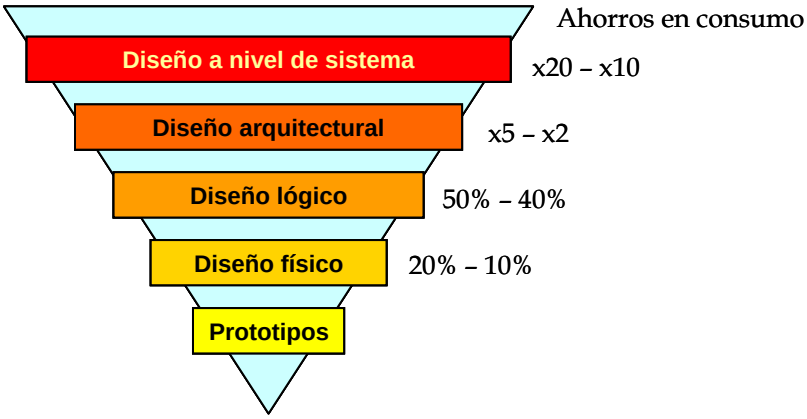


Figura 1-4: Capacidad para influir en el consumo según las etapas del diseño

En la etapa de **diseño a nivel de sistema**, se exploran las distintas arquitecturas posibles y los beneficios e inconvenientes respecto a las prestaciones y requisitos del sistema. Por ejemplo, para esta etapa ya se habrían valorado las ventajas de la implementación del circuito en un ASIC, en vez de en un dispositivo programable, como lo sería una FPGA.

[9] Por claridad se usará el término "dopado" de origen anglosajón en vez de "contaminación" ya que su uso está más extendido

Como ejemplos, en esta etapa se determina si el diseño va a estar basado en microprocesador o se va a emplear un hardware específico; cómo se realizará la partición hardware/software; y qué módulos se diseñarán y en cuáles se recurrirá a la reutilización y el empleo de componentes de propiedad intelectual (IP).

Estas decisiones limitan enormemente el rango de actuación de las siguientes etapas. El resultado de esta etapa es un modelo del circuito a nivel comportamental, en el que se especifica la funcionalidad, las características y las interacciones entre los módulos y el exterior. Las interacciones se realizan a nivel de transacciones (TLM[10]), en las que las sincronizaciones no se basan en los ciclos de reloj, sino en la comunicación de datos entre bloques [14]. En este nivel, se emplean lenguajes de alto nivel como por ejemplo UML, SystemC, SystemVerilog, o incluso, aunque con más limitaciones, VHDL.

Durante el **diseño arquitectural** se persigue obtener un modelo en nivel de transferencia de registros (RTL[11]) del circuito, normalmente descrito en un lenguaje de descripción de hardware como VHDL o Verilog. Cada módulo, además de cumplir con la funcionalidad especificada en la etapa anterior, deberá satisfacer las restricciones impuestas en las especificaciones. Para circuitos síncronos, se obtiene una descripción compatible en ciclos de reloj con el circuito real, aunque sin tener en cuenta los retardos de la lógica combinacional.

Ejemplos de las optimizaciones que se realizan en este nivel son la elección de la descripción más eficiente para cada módulo, paralelizar o multiplexar sub-módulos, adoptar estructuras segmentadas (*pipeline*), etc.

La etapa de **diseño lógico** está fuertemente automatizada, en ella se obtiene el listado de puertas y sus conexiones (*netlist*). Ejemplos de técnicas de optimización que forman parte de esta etapa son: reducción de transiciones espurias, *puerteado* de relojes, inhabilitación de elementos de memoria, optimización de la codificación, organización óptima del árbol de reloj, etc.

Por último, en la etapa de **diseño físico** se obtiene la topografía del circuito (*layout*) con la geometría de los componentes y sus conexiones. En esta etapa, las posibilidades de reducción de consumo se basan en la elección de tensiones de alimentación V_{dd}, tensiones umbral V_T, y dimensiones de los transistores; además de multitud de técnicas específicas para reducir el consumo estático [137], como por ejemplo, tecnologías de tensión umbral dual [24], tensiones umbral múltiple [89], tensión umbral variable [72], tensión umbral dinámica [3], tensiones de alimentación múltiples [135].

Es importante señalar que no siempre es posible encasillar las técnicas en un único apartado, existiendo técnicas que pueden ser aplicadas en varios niveles, como son la optimización de la codificación, reducción de transiciones espurias, *puerteado* e inhabilitación entre otras.

1.3.3 Consumo por computación y consumo por pérdidas

En esta tesis doctoral, el trabajo está orientado a los circuitos integrados digitales de procesamiento, y no a los de almacenamiento (memorias). Estos circuitos realizan un procesamiento de las entradas[12] para proporcionar una determinada información a la salida.

Para realizar el cómputo, el circuito realiza operaciones mediante interruptores, formados por transistores, que van cambiando el estado de las señales internas hasta llegar a las salidas. Los cambios de estado de las señales del circuito suponen un **gasto energético para poder llevar a cabo la conmutación** que carga o descarga un condensador de control, que se encarga de mantener la información [58]. Este proceso de carga y descarga del condensador de control se corresponde con la carga y descarga de las capacidades internas de los transistores descritas en

[10] Del inglés: *Transaction Level Modeling*
[11] Del inglés: *Register Transfer Level*
[12] Y del estado interno si es un circuito secuencial

la sección 1.3.1.1, en la que el condensador de carga C_L equivale al condensador de control (figura 1-2), ya que el condensador de carga representa las capacidades internas de los transistores.

Así pues, la **computación implica un gasto energético**, cuyo mecanismo está regido por la carga y descarga de las capacidades internas explicadas en la sección 1.3.1.1. y por tanto, existe un **consumo debido a la computación**, que está ligado a la **actividad de conmutación** [44].

Sin embargo, se debe señalar que hay opiniones enfrentadas acerca de la posibilidad de eliminar este consumo mediante la hipótesis de la computación reversible y la conmutación adiabática [9], [58].

Sin entrar en detalle, la discusión se centra en los límites energéticos de los interruptores binarios electrónicos para el procesamiento de la información. Por un lado, unos afirman que una transición binaria, como acto elemental de procesamiento de información, siempre requiere una energía mínima, cuyo valor se puede obtener a partir de los trabajos de la teoría de la comunicación de Shannon [123], y extendidos a la computación por von Neumann (pág. 66 [136]). Por otro lado, la otra corriente (Landauer [74] y Bennett [10] entre otros) afirma que la comunicación y computación se puede hacer con tan poca disipación por bit como se quiera siempre que no se elimine información.

Independientemente de la anterior discrepancia, en los circuitos electrónicos actuales, la viabilidad de la hipótesis de carga adiabática de las capacidades internas es debatible[13] [58], y si acaso, sería más viable para otras tecnologías, como las relacionadas con el área de la computación cuántica. Así pues, en esta tesis se asumirá la existencia y necesidad[14] de un consumo por computación.

Aparte de este consumo por computación, existen **pérdidas no imprescindibles** de energía debidas a las características no ideales de los dispositivos. Una de ellas es el **consumo estático** (sección 1.3.1.3), ligado al escalado no ideal de los transistores y asociado al mantenimiento de los niveles lógicos entre conmutaciones, su existencia supone un desperdicio energético por no contribuir a la computación [44]. Este consumo hace que la lógica digital (el área de silicio) tenga un coste energético, e introduce un parámetro más en la optimización del circuito [82]. Ya se ha señalado además, lo crítico que esta siendo su aumento para el desarrollo de la microelectrónica [63].

También hay **pérdidas energéticas asociadas a la conmutación** que no realizan trabajo. Una de ellas es el consumo por corrientes de cortocircuito (sección 1.3.1.2), que se debe al comportamiento no ideal de las transiciones de las señales. La pendiente finita de las transiciones de las señales a la entrada de los transistores hace que se pierda energía al estar los transistores complementarios en conducción (figura 1-3).

Una causa importante en el consumo dinámico se debe a las **transiciones espurias** (*glitches*), que según el circuito y las condiciones de funcionamiento pueden llegar a constituir un 40% ó más del consumo total del circuito [107]. Estos pulsos no deseados se deben a los retardos de las puertas lógicas, que producen diferencias en los tiempos de llegada de las señales. En la figura 1-5 se muestra la formación de una transición espuria causada por la llegada con retraso de la señal -*B*, debido al retardo introducido por el inversor.

[13] Tanto por las pérdidas energéticas que llevaría el uso de una fuente de alimentación especial, como por el efecto de los errores debidos a las transiciones lentas, y debido a que las ventajas de la carga adiabática se ven eclipsadas por la lentitud del proceso

[14] Este consumo es necesario debido a que es inevitable al realizar la computación. Y por tanto, al contrario que el resto del consumo energético, éste no sería estrictamente una pérdida sino un gasto

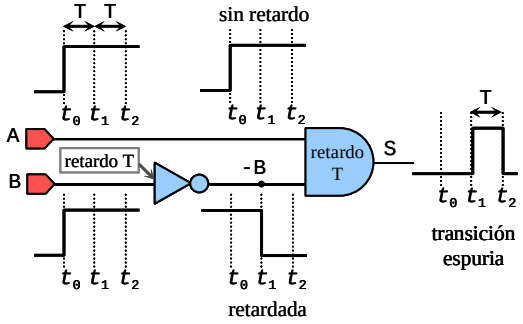


Figura 1-5: formación de transiciones espurias

El **consumo debido a las transiciones espurias es inútil**, ya que no realizan cómputo alguno. La cantidad de energía disipada se calcula de manera análoga a la explicada para el consumo dinámico (secciones 1-2 y 1-3), aunque debido a que no siempre las transiciones son completas, como la de la figura 1-6, el consumo de una transición de este tipo equivaldría a una fracción del total (ecuación 1.11)

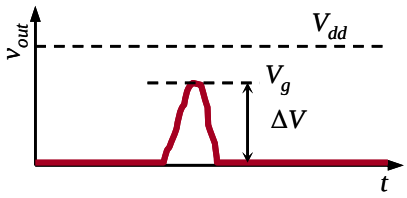


Figura 1-6: gráfica de una transición espuria incompleta

La energía consumida en este tipo de transiciones incompletas se puede aproximar a [96]:

$$E_{Vdd} = C_L \cdot V_{dd} \cdot \Delta V \tag{1.11}$$

Una vez generados estos pulsos espurios, se propagan por la parte combinacional del circuito, incrementando la actividad del circuito. Los pulsos estrechos sufren una degradación en la propagación, resultando que muchas de estas transiciones terminan desapareciendo [113].

La generación de transiciones espurias depende de la implementación de la función lógica. Esto se puede observar en la figura 1-7, donde con las mismas entradas, la estructura en cadena, con caminos desequilibrados, genera más transiciones espurias que en la estructura en árbol.

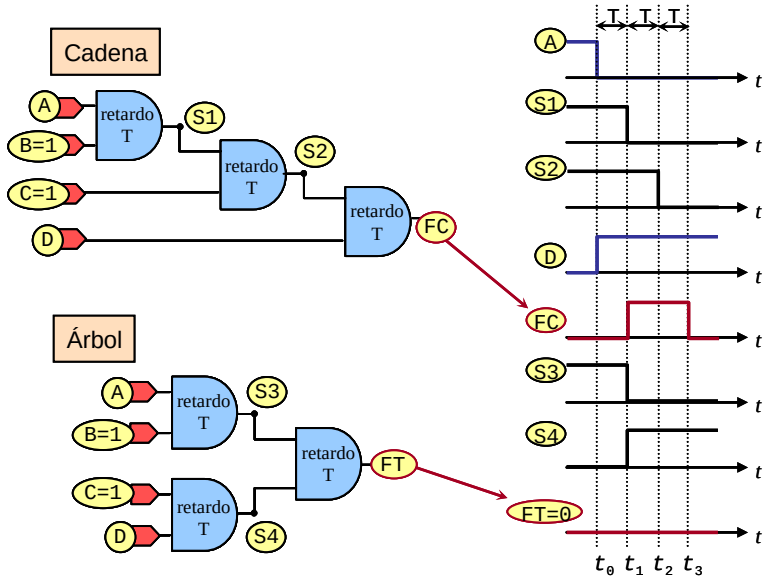


Figura 1-7: diferencia de la estructura en cadena y árbol para la generación de espurios

1.4 La actividad de conmutación

La estimación del consumo no es el único campo de aplicación de la actividad de conmutación, de hecho, antes de que el consumo fuese un aspecto tan significativo en los circuitos digitales, fueron realizados estudios sobre la actividad de conmutación para analizar la **fiabilidad** [93].

En el campo del análisis de la **testabilidad** y computación con tolerancia a fallos de los circuitos, muchos trabajos se han realizado analizando las probabilidades de señal [103], [140], [69], [83], [37], [71], [38], que, como se verá en el capítulo siguiente, en muchas ocasiones constituye un paso previo para el análisis de la actividad de conmutación.

En la sección 1.3.1.3 se explicó la dependencia de las corrientes subumbral con la combinación de valores a las entradas de las puertas [25], [51], [40], [100]. Las probabilidades de señal pueden servir de ayuda para estimar los valores de las entradas de las puertas y así estimar y disminuir el consumo estático.

La **actividad de conmutación es una medida del número de transiciones que experimentan las señales de un circuito**. Por lo tanto es una medida del comportamiento del circuito, y en cierta manera, de la información que procesa y transporta. El comportamiento de un circuito es variable, dependiendo de las señales de entrada que reciba y del estado interno (en caso de que sea un circuito secuencial). Por tanto, la actividad de conmutación de un circuito deberá estar asociada a una secuencia o patrón de las entradas. Además, para el caso de circuitos secuenciales, también podrá estar asociada a un modo de funcionamiento del circuito.

En consecuencia, la actividad tiene dos aspectos: uno relativo al circuito, que se puede relacionar con la capacidad computacional y de procesamiento del circuito; y el otro relacionado con las entradas, que se refiere a la información que tiene que procesar. Así, puede ocurrir que un circuito con muchas capacidades de cálculo reciba como entradas algo muy sencillo de procesar; y por lo tanto, para estas condiciones el circuito tendrá una baja actividad.

La comparación entre la actividad de conmutación de distintas zonas del circuito permite al diseñador conocer dónde está la mayor carga de procesamiento, permitiéndole además **verificar** si se corresponde con lo esperado para las condiciones que se imponen. La detección de comportamientos anómalos de señales puede conducir a la **localización de errores** de diseño del circuito. Y la comparación de probabilidades y actividades de conmutación entre las salidas de distintas implementaciones de un circuito puede permitir verificar si realmente realizan la misma función. Así, la medida de la actividad ofrece una **exploración del comportamiento** del circuito.

La existencia de señales con muy baja actividad no tiene necesariamente que implicar un error en el diseño, pero puede ser una muestra de poca **controlabilidad**, lo cual debe ser tenido en cuenta si se quiere realizar un diseño acorde con las metodologías de test (DFT[15]) [38].

Por último, la **fiabilidad** del circuito se compromete en circuitos de actividad elevada debido a que la disipación energética eleva la temperatura del circuito. Estas altas temperaturas, no sólo hacen que el circuito funcione incorrectamente por el aumento de los retardos, sino que pueden dañar el circuito permanentemente.

Además, la fiabilidad no sólo se ve comprometida por el aumento de la temperatura, sino que para circuitos de señal mixta (digitales y analógicos), la actividad de conmutación de los bloques de circuitos digitales inyectan corrientes que producen ruido de alta frecuencia, tanto a la red de alimentación y tierra, como al substrato de los circuitos integrados. Estas corrientes pueden producir electromigración, caídas de tensión óhmica (*IR drops*), oscilaciones de tensión debidas a resonancias y posiblemente interferencias electromagnéticas [90], [84], [104], [94], [143].

[15] *Design for Testability*: metodologías de diseño que facilitan las tareas de test del circuito

1.5 Conclusiones

En este capítulo se ha esquematizado el proceso del diseño electrónico digital y se han introducido los principales problemas a los que se enfrenta. La necesidad de metodologías y herramientas de ayuda al diseño ha quedado patente por la creciente complejidad de los circuitos y procesos tecnológicos. Las metodologías y herramientas de ayuda al diseño no sólo deben asistir en cuanto al correcto funcionamiento del circuito, sino que además, deben de ayudar en el cumplimiento de ciertas características adicionales, como lo son las prestaciones, el consumo, el área,....

En consecuencia, son fundamentales herramientas que estimen este tipo de características, ya que un conocimiento anticipado de estas características puede ayudar a evitar iteraciones en el flujo del diseño. Estas iteraciones ocasionan gravosos retrasos en la salida del producto y hacen aumentar los costes de producción.

Además, la **optimización** de estas características de un circuito **es más eficaz cuanto más alto sea el nivel de abstracción,** ya que se cuenta con un mayor número de grados de libertad. Sin embargo, a alto nivel, al no estar aún definidos aspectos de detalle del circuito, hay una mayor distancia entre la descripción del diseño y el circuito final. En consecuencia, **son necesarios estimadores** que acerquen al diseñador al circuito final, que le permitan conocer si se cumplen las restricciones y que le ofrezcan un medio de comparación entre distintas implementaciones.

Dentro de las características de los circuitos digitales, el consumo tiene una importancia notoria. La amplia difusión de los dispositivos portátiles, así como los altos niveles de disipación energética de los circuitos de altas prestaciones, han hecho del **consumo una prioridad en el diseño**.

La **actividad de conmutación es un factor fundamental en el consumo** de los circuitos, ya que influye decisivamente en el consumo dinámico. Además, el estudio de la actividad de conmutación tiene otras aplicaciones, como lo son análisis de testabilidad, fiabilidad y calidad del circuito, así como un medio para la exploración del circuito.

El estudio de la actividad de conmutación es el objetivo de esta tesis. En esta tesis se propone un método probabilístico para estimar la actividad de conmutación. La novedad de la propuesta radica en que el método propuesto se lleva a cabo en el nivel RT. Antes de proceder a explicar la propuesta de estimación de la actividad desarrollada en esta tesis (capítulo 3), en el capítulo siguiente se expondrán las diferentes alternativas de estimación aportadas por la comunidad científica.

2. LA ESTIMACIÓN DE LA ACTIVIDAD DE CONMUTACIÓN

En este capítulo se exponen los métodos de estimación de la actividad de conmutación de diseños digitales. Hay dos maneras de afrontar la estimación de la actividad, una utiliza la simulación (método dinámico), y la otra emplea modelos analíticos (método estático). En el apartado 2.1 se profundiza en estos métodos, analizando sus diferencias y las ventajas de cada uno.

Por las razones que se exponen en el apartado 2.1, en esta tesis se ha optado por el método estático. A causa de que estos métodos requieren un modelado y desarrollo matemático específico, además de una terminología asociada, la sección 2.2 se dedica completamente a establecer esta terminología y los fundamentos matemáticos que se emplean.

La sección 2.3 se hace una revisión de las distintas propuestas en los métodos probabilísticos, extrayendo las aportaciones más importantes que se han ido formulando.

Para terminar este capítulo se encuadrará el trabajo de esta tesis doctoral, señalando aquellos aspectos que pretende cubrir.

Se debe de tener en cuenta que la mayoría de las propuestas descritas tienen el objetivo final de estimar el consumo. Por ello, en muchas ocasiones aparecerán referencias al consumo y su terminología asociada.

2.1 Métodos de estimación de actividad de conmutación

El cálculo de la actividad de conmutación entraña gran dificultad debido a que no hay un valor fijo de actividad asociado a cada circuito, sino que depende tanto de la estructura y funcionalidad del circuito, como de la tarea que se esté desempeñando en ese momento.

Los métodos para la estimación de la actividad de conmutación se dividen en [91], [128]:

- Dinámicos, también llamados simulativos o estadísticos
- Estáticos, también llamados analíticos o probabilísticos

Los métodos dinámicos extraen las estadísticas de la actividad de conmutación mediante la **simulación** del circuito. Los métodos estáticos emplean **métodos analíticos** para crear un modelo del circuito por el que se propagan los valores de probabilidad de señal y actividad de conmutación.

El esquema general de ambos enfoques se muestra en la figura 2-1. Los métodos dinámicos simulan el circuito con un largo número de patrones de entrada y, observando los datos estadísticos de la simulación, extraen la actividad y probabilidad de las señales del circuito. Debido a que los tiempos de simulación resultan muy largos para circuitos grandes, muchas de estas técnicas recurren al análisis de los vectores de entrada para reducir su tamaño.

Por otro lado, la técnica probabilística, realiza un único análisis del circuito a partir de el cual obtiene un modelo probabilístico. Este modelo permite propagar rápidamente los valores de probabilidad y actividad por todo el circuito.

Simplificando, se podría decir que los métodos simulativos son extensivos en tiempo, es decir, estos métodos no realizan grandes esfuerzos computacionales, sino que es el simulador lleva la mayor carga. Por otro lado, los métodos estáticos son intensivos, requiriendo grandes esfuerzos computacionales pero solucionando el cálculo en mucho menos tiempo.

En los siguientes subapartados se detallarán ambos métodos, para posteriormente valorar sus ventajas e inconvenientes.

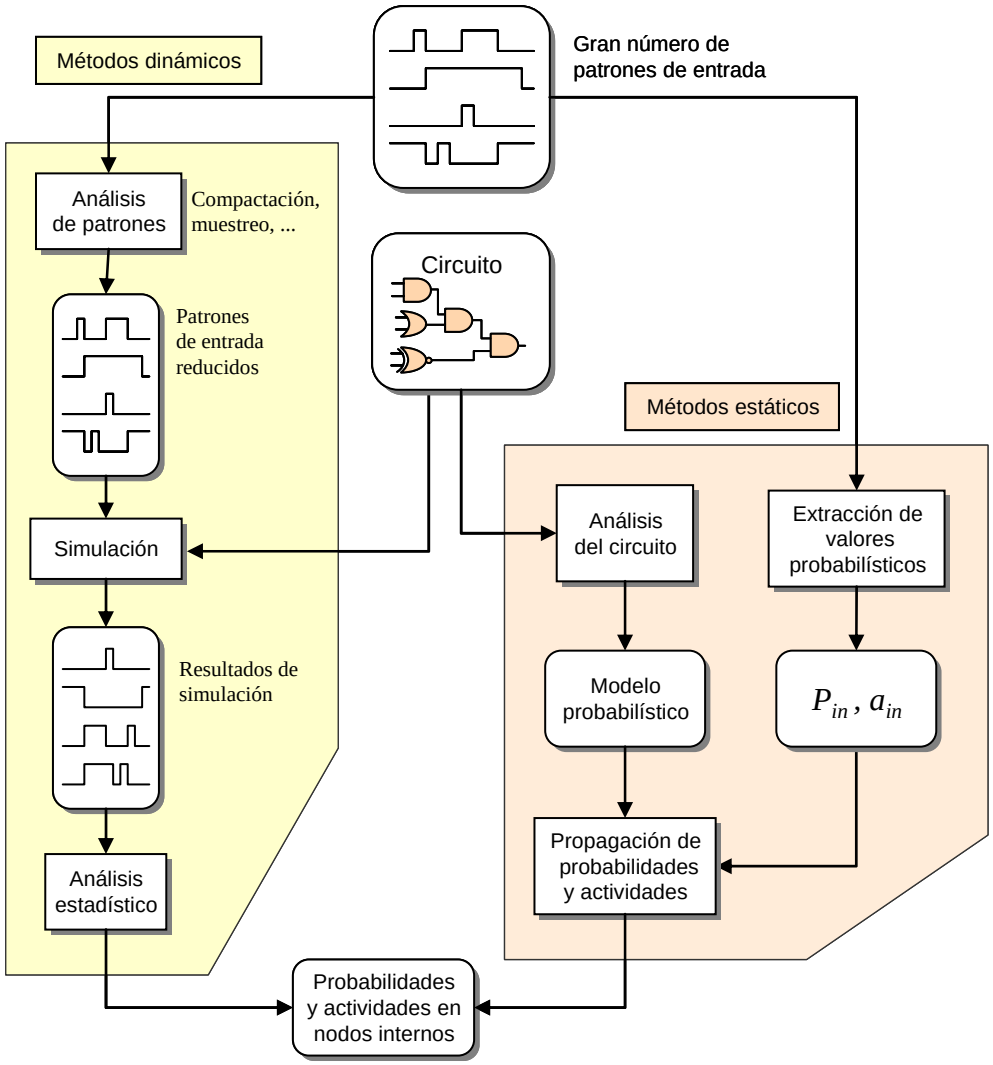


Figura 2-1: Esquema de los métodos dinámicos y estáticos para la estimación de la actividad

2.1.1 Métodos dinámicos

Las técnicas dinámicas, también llamadas estadísticas o simulativas, recurren a la simulación para obtener la actividad de las señales internas de un circuito. La observación de la simulación de un circuito con unas condiciones de entrada determinadas proporciona el conocimiento de su actividad. Los inconvenientes de este tipo de análisis son los tiempos empleados en la simulación y lo limitado de sus resultados, ya que el mismo circuito en otro banco de pruebas podría dar una actividad totalmente diferente. Ejemplos de este análisis es el realizado por Dresig [35], o la metodología propuesta en el *Power Compiler*™ de *Synopsys*® [126] para la estimación del consumo.

Así, en la metodología propuesta en el *Power Compiler* se realiza una simulación con retardos del circuito en nivel de puertas procurando que sea con un banco de pruebas que se ajuste lo máximo al funcionamiento real. De la simulación se extraen las probabilidades y actividades de señal de todas las señales del circuito, en un fichero en formato SAIF[16]. El análisis del consumo se realiza a partir de estos datos de actividad, junto con la información de la disposición de las puertas del circuito (*netlist*) y la biblioteca tecnológica que aporta la información del consumo.

[16] SAIF: *Switching Activity Interchange Format*. Formato propuesto por *Synopsys* para intercambiar información de la actividad de conmutación

Debido a que los tiempos de simulación con retardos en el nivel de puertas pueden ser muy grandes, proponen como alternativa obtener los datos de actividad y probabilidad mediante simulación en RTL sin retardos. Como en RTL la lógica combinacional está sujeta a cambios después de la síntesis, de la simulación se recogen solamente los datos de las señales invariantes (señales primarias), es decir puertos y elementos de memoria. Posteriormente, mediante simulación sin retardos, la herramienta propaga esos valores por la lógica combinacional ya sintetizada en puertas. El simulador emplea la funcionalidad de los bloques y vectores aleatorios para esta propagación, el número de pasos de la simulación depende del esfuerzo requerido por el usuario.

En la figura 2-2 se muestra el esquema de la metodología propuesta, en el flujo de la izquierda se sigue la simulación RTL; en el de la derecha se realiza la simulación en nivel de puertas.

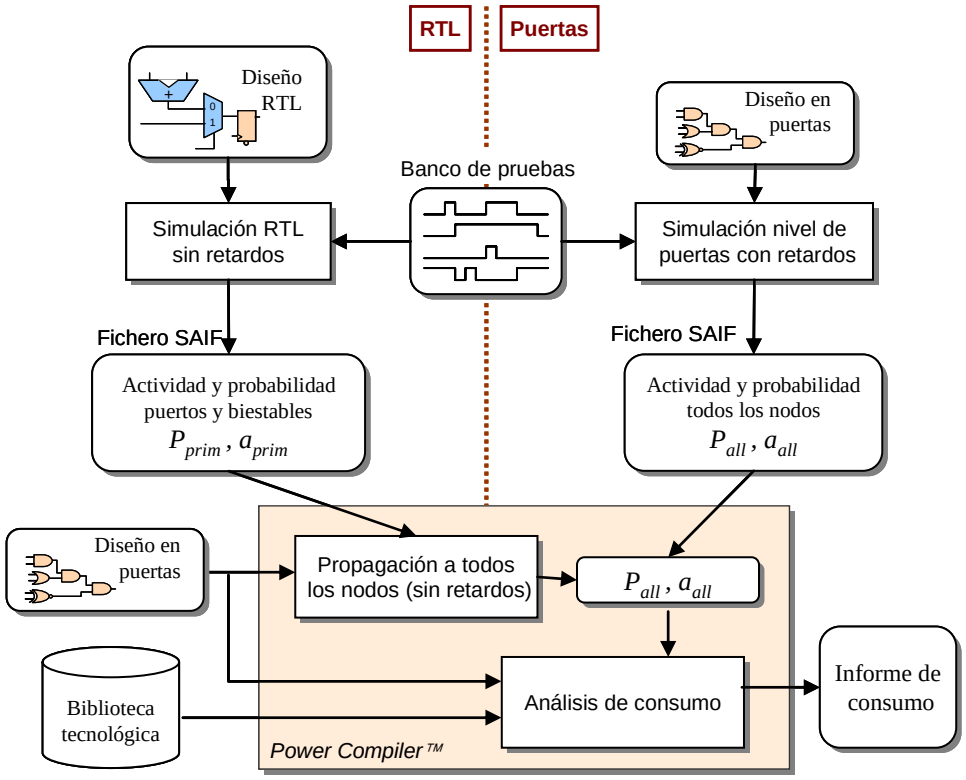


Figura 2-2: Metodología para la estimación de consumo propuesta en la herramienta Power Compiler de Synopsys

Para acortar los tiempos de simulación y asociar los resultados de actividad a unas condiciones de entrada, Burch [18] propone una **simulación de Monte Carlo** implementada en el programa McPower. Este enfoque consiste en la simulación del circuito mediante la aplicación de vectores de entrada aleatorios con una determinada probabilidad y actividad. El circuito es analizado durante la simulación, calculándose su consumo total. La simulación se detiene cuando la variación de consumo calculado durante un intervalo de tiempo no supera un porcentaje de error y está dentro de un nivel de confianza determinado. Para este método, la convergencia es rápida si se considera el consumo total, pero si se pretende que la convergencia se produzca para cada señal interna y no a nivel global, la convergencia es muy lenta.

Para evitar este problema Xakellis, implementando la herramienta MED [141], propone que la convergencia en las señales con baja actividad se haga en términos absolutos, ya que son los que hacen que la convergencia sea lenta, y son señales con menor influencia en el consumo y en problemas de electromigración. MED también considera circuitos secuenciales pero la simulación puede verse atrapada en un conjunto de estados muy probables y hacer que, sin haber salido de estos estados, erróneamente parezca que se ha alcanzado la convergencia.

Chou [27] solventa esta cuestión con la herramienta MCS, en donde lleva a cabo una simulación de calentamiento para identificar aquellos estados próximos y las probabilidades de alcanzar

los conjuntos de estados próximos. Así, se realiza la simulación de Monte Carlo asegurando que se alcanzan todos los conjuntos de estados próximos.

Aún así, según Ding [34] el método de Monte Carlo tiene tres inconvenientes:

- Requiere vectores de test largos para algunos casos
- Las correlaciones espacio-temporales no se consideran adecuadamente
- El criterio de parada asume que las entradas siguen una distribución normal. Cuando esto no ocurre, la simulación podría terminar prematuramente y, en consecuencia, no se alcanzarían todos los conjuntos de estados próximos

Por tanto, Ding propone realizar la estimación mediante **muestreo aleatorio estratificado** [34]. En éste se divide la población (los elementos de los vectores de test) en grupos disjuntos, y éstos se agrupan en características de consumo similares, que son calculadas mediante un predictor. Posteriormente, a cada estrato se le aplica un muestreo aleatorio simple, obteniendo el consumo medio mediante simulación de un número de muestras hasta que se cumple un criterio de parada. Este método plantea algunas dificultades en la elección de los estratos y la asignación de las muestras en los estratos. Por otro lado, cuando la población es muy grande, el coste del cálculo del predictor llega a ser importante. Sus resultados experimentales muestran mejoras de tiempos respecto a la simulación de Montecarlo de entre 3 y 10 veces según el tipo de secuencias de entradas.

Otra propuesta para disminuir la longitud de los vectores de test es la **compactación de vectores**, propuesta por Marculescu [81]. El objetivo de ésta es disminuir la longitud de los vectores de entrada preservando sus propiedades estadísticas (correlaciones espacio-temporales de primer orden). El método modela los vectores de entrada como cadenas de Markov dinámicas, y para capturar las dependencias temporales de orden superior recurren al uso de árboles de Markov dinámicos de orden *k*.

En otra propuesta para disminuir la longitud de los vectores, Macii [77] propone la **síntesis de vectores** en vez de la compactación. La diferencia estriba en que todas las tramas generadas en la compactación están incluidas en la secuencia original, mientras que la síntesis puede producir patrones que no aparecen en la secuencia original. La síntesis propuesta se basa en el análisis de las propiedades espectrales de la actividad de los vectores de entrada.

La figura 2-3 muestra un resumen de las técnicas simulativas que se acaban de describir.

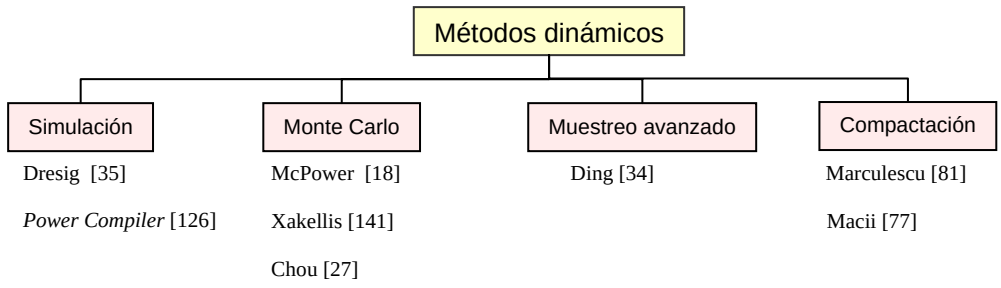


Figura 2-3: Esquema de técnicas dinámicas en el nivel de puertas

Por último, debido a que los métodos dinámicos requieren repetir todo el proceso cuando se modifican las condiciones de las entradas, se han realizado propuestas que elaboran modelos a partir de un conjunto de simulaciones con distintas condiciones de entrada. Con estos modelos se evita tener que repetir la simulación con cada cambio en las entradas. Por ejemplo, Bernacchia propone elaborar modelos analíticos a partir de las estadísticas de actividad del circuito [12]. Mientras que Gupta opta por construir tablas con las que, mediante interpolación, obtienen los resultados de actividad [49].

2.1.2 Métodos estáticos

Los métodos estáticos calculan la actividad de las señales de un circuito sin recurrir a la simulación. Realizan un análisis del circuito con el que elaboran un modelo probabilístico mediante el cual propagan los valores de probabilidad y actividad de las entradas por todo el circuito. Por ello, también se conocen como métodos **analíticos** o **probabilísticos**.

Para la elaboración del modelo probabilístico, a cada puerta lógica se le asocia una ecuación de probabilidad y actividad que relaciona la probabilidad y actividad de la salida con las de las entradas de la puerta. En la figura 2-4 se muestra el fundamento de la obtención de la ecuación de probabilidad para la puerta AND. La puerta AND requiere que ambas entradas, *B* y *C*, sean '1' para que la salida *D* sea '1'. Por tanto, $P_D = P\{(B=1) \wedge (C=1)\}$. Donde $\wedge$ es el operador AND-lógico. Si se supone que las señales *B* y *C* son mutuamente independientes (§2.2.1), la expresión de probabilidad se puede separar, resultando $P_D=P_B \cdot P_C$.

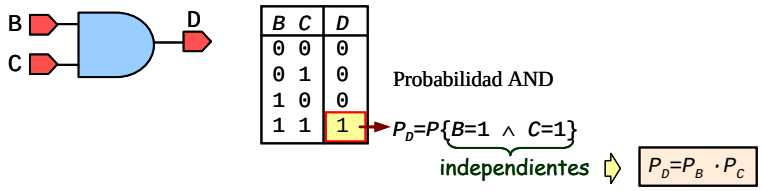


Figura 2-4: Extracción de la fórmula de probabilidad para la puerta AND

El proceso es similar para la obtención de la ecuación de la actividad de la puerta AND, con la diferencia de que al considerar las señales en dos tiempos consecutivos, el número de variables se multiplica por dos y, por tanto, el número de sucesos posibles se eleva al cuadrado (véase la figura 2-5).

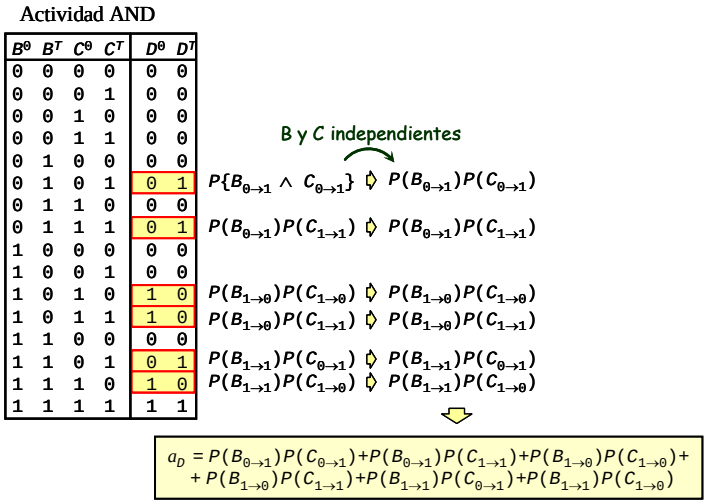


Figura 2-5: Extracción de la fórmula de actividad para la puerta AND

Las ecuaciones resultantes de probabilidad (figura 2-4) y actividad (figura 2-5) se han dejado sin simplificar en la forma de suma canónica de productos. Independientemente de que se hubiesen expresado en formas más compactas, se puede apreciar la complejidad relativa de la ecuación de la actividad, incluso para ejemplos sencillos como el de la puerta AND.

La ecuación de probabilidad de la puerta OR se obtiene de manera similar, extrayendo los sucesos que resultan en la condición de salida deseada. Para este caso se observa que la probabilidad de la salida se forma a partir de tres términos de la tabla de verdad $\{\{A=0 \wedge D=1\}, \{A=1 \wedge D=0\}, \{A=1 \wedge D=1\}\}$. Cada uno de estos términos (filas) de la tabla de verdad se corresponde con un suceso disjunto del resto, y por tanto sus probabilidades se pueden calcular separadamente (ecuación 2.8). En la figura 2-6 se muestra la obtención de la ecuación de probabilidad de la puerta OR y la ecuación resultante de la actividad (obtenida directamente, sin incluir la tabla de verdad como se hizo en la figura 2-5).

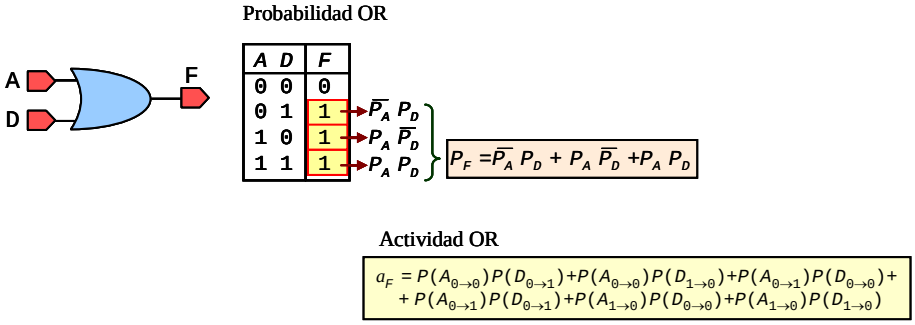


Figura 2-6: Extracción de las fórmulas de probabilidad y actividad para la puerta OR

Con el fin de clarificar el proceso de propagación, en la figura 2-7 se ha añadido un ejemplo numérico. Los puertos de entrada tienen asignada una probabilidad y actividad que, mediante las fórmulas deducidas anteriormente, se han propagado numéricamente a las señales *D* y *F*.

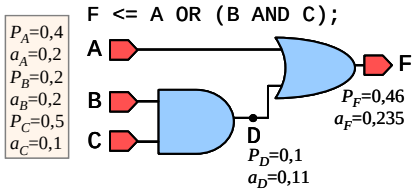


Figura 2-7: Propagación de probabilidades y actividades para un circuito simple de ejemplo

Como se justificará en la sección 2.1.3, ésta es la técnica empleada en esta tesis y por tanto, se dedicará la sección 2.2.3 a profundizar en las dificultades del método y la sección 2.3 a resumir las distintas soluciones que se han plateando.

La dificultad del análisis probabilístico depende del tipo de circuito, la manera de describirlo y las consideraciones que se hagan. Así, por ejemplo, los circuitos secuenciales y con reconvergencias de señales entrañan mucha más dificultad. Igualmente, considerar modelos con retardos o con correlaciones espacio-temporales en las entradas incrementa mucho el coste computacional del método. Sin embargo, no tener en cuenta algunos de estos aspectos puede llevar a una baja fiabilidad del método. Por tanto requieren de un compromiso entre complejidad y fiabilidad.

2.1.3 Comparación entre los métodos dinámicos y estáticos

No existe un método que sea claramente superior al otro, ya que ambos tienen inconvenientes significativos. En el caso de los métodos dinámicos, su punto débil más importante es el tiempo empleado en la simulación, que para circuitos grandes lo hace inviable. Por esto surge la necesidad de emplear técnicas que disminuyan el tamaño de los patrones de test de las simulaciones. A este inconveniente se le añade la dependencia de los resultados con los patrones de test empleados en la simulación y la necesidad de repetir todo el proceso cuando las condiciones de entrada varían, o en otro caso, recurrir a la interpolación entre varias condiciones conocidas o hacer modelos de regresiones.

Estos inconvenientes no los tienen los métodos estáticos, pues los resultados no dependen de unos patrones de test, debido a que directamente se propagan los valores de probabilidad y actividad. Es por tanto un método teórico que no depende de cada caso particular. En principio, los métodos estáticos son rápidos, aunque según las consideraciones tomadas acerca del circuito, la elaboración del modelo puede llegar a tomar mucho tiempo o incluso a ser inabordable para circuitos muy grandes. Sin embargo, tienen la gran ventaja de que, una vez elaborado el modelo probabilístico, si se varían las condiciones de entrada, simplemente se propagan las probabilidades sobre el mismo modelo, siendo considerablemente más rápido que el dinámico.

No obstante, los métodos estáticos deben realizar simplificaciones a la hora de construir el modelo, ya que en otro caso se requerirían muchos recursos computacionales y de memoria, e incluso podrían hacer su realización inviable. Por tanto, la ventaja de su independencia con los vectores de entrada puede verse mermada por la pérdida de fiabilidad debido al uso de un modelo que difiera mucho de la realidad.

En los modelos dinámicos esto no ocurre, ya que se realiza la simulación directamente con el diseño a estimar. Esto proporciona exactitud y fiabilidad, y una gran independencia de los métodos dinámicos respecto al tipo de descripción, aunque, indudablemente, el estilo de descripción influye en los tiempos de simulación. Por otro lado, la elaboración de los modelos probabilísticos sí que tiene una fuerte dependencia con el tipo de descripción del circuito.

Los métodos estáticos son más directos en su uso, pues no precisan de la elaboración de un banco de pruebas como ocurre con los dinámicos. Por tanto son más adecuados durante la tarea del diseño, ya que incluso cuando todavía no está terminado se puede analizar la actividad.

El análisis estático proporciona además información complementaria que puede ser muy valiosa. Ya que las relaciones e influencias entre las señales pueden dar a conocer las causas de la actividad de determinadas señales y áreas del circuito. Por otro lado, este mismo análisis puede ayudar a sacar a la luz posibles errores en el diseño y descripciones inadecuadas, con lo que contribuye al diseño de un circuito de mayor calidad.

A modo de resumen, la tabla 2-1 compara las características generales de ambos métodos.

Métodos dinámicos (simulativo)	Métodos estáticos (probabilístico)
☹ Tiempos largos de simulación	☹ Complejidad en análisis
☹ Procesamiento para cada situación en entradas	☺ Procesamiento de una vez
☹ Interpolación de resultados	☺ Rápido al cambiar las condiciones de entrada
☺ Exacto y fiable	☹ Complejidad depende del estilo de descripción
☺ Complejidad independiente del estilo de descripción	☹ Grandes requerimientos computacionales
☹ Dependiente de la secuencia de entrada	☹ Necesidad de simplificaciones y particiones
☹ Tener diseño y banco de pruebas terminado	☺ Se puede realizar en diseño inacabados
	☺ Análisis útil para exploración del circuito

Tabla 2-1: Características generales de los métodos dinámicos y estáticos

Como ya se ha comentado, no hay una técnica claramente ventajosa sobre la otra y ambas se siguen desarrollando por diferentes autores. De todas maneras, en la actualidad hay una tendencia hacia los métodos estáticos debido a que el tamaño de los diseños actuales hace que los tiempos requeridos para los métodos simulativos sean excesivamente largos [47].

Asimismo, la exploración del diseño necesaria para la técnica estática resulta muy útil como complemento y continuación de la línea de trabajo realizado en el grupo de investigación (CEI[17]) donde se desarrolla esta tesis doctoral. Por esto, la herramienta de ayuda al diseño Ardid [132] y su Modelo Simplificado del Hardware (SHM) [131] se ven beneficiados por las aportaciones para realizar estimaciones y evaluar la calidad del diseño. A la vez que la elaboración del modelo probabilístico resulta más sencilla por la experiencia y el desarrollo previo estas de herramientas.

Así pues, las razones fundamentales que han llevado a la elección del método probabilístico son:

- Evitar los largos tiempos de simulación: tendencia por los métodos estáticos.

[17] CEI: Centro de Electrónica Industrial, de la Universidad Politécnica de Madrid (UPM)

- Continuación de la línea de investigación del departamento relativa al análisis estático y exploración del circuito. Esto ofrece la experiencia acumulada en este campo y se continúa aportando en el mismo sentido.
- Que se pueda realizar en diseño inacabados, ya que permite realizar la estimación antes en el flujo del diseño, y sin necesidad de construir el banco de pruebas.

Es por todo esto que se ha escogido el método probabilístico, no obstante, en las líneas de investigación del departamento también se realizan estudios relativos a las técnicas dinámicas [36].

A raíz de esta elección, los apartados 2.2 y 2.3 profundizan con las técnicas probabilísticas y las distintas aportaciones realizadas.

2.2 Fundamentos de la estimación probabilística de la actividad de conmutación

La estimación probabilística de la actividad de conmutación implica la elaboración de un modelo matemático del circuito. Su complejidad dependerá de las características del circuito y de las consideraciones y simplificaciones que se tomen para el modelo.

En este capítulo se describirá el método probabilístico y sus dificultades más notables. Con anterioridad se expondrá la terminología y definiciones empleadas en este modelo, dedicándose también un apartado a exponer los diagramas de decisión binaria (BDD), ya que son ampliamente utilizados para el cómputo de la probabilidad y actividad, y son una de las bases del desarrollo de esta tesis doctoral. Por último, se repasarán las distintas aportaciones que se han propuesto, analizando que aspectos no están cubiertos y señalando aquellos que esta tesis pretende cubrir. Estas aportaciones se exponen después de las secciones relativas a la terminología y los BDD debido a que las aportaciones hacen referencia a la terminología y los conceptos de estas secciones.

2.2.1 Definiciones y terminología

En los estudios probabilísticos se asocia cada señal del circuito con una variable aleatoria modelada con proceso estocástico estrictamente estacionario, siendo {0, 1} el conjunto de valores que la variable puede tomar. Para simplificar la notación, la señal y la variable aleatoria se denominarán igual.

A continuación se presenta la terminología y definiciones asociadas al modelo probabilístico:

- *Probabilidad de señal*, P_x , es la probabilidad de que la señal x valga 1. Las siguientes representaciones son equivalentes:

  $$P_x \equiv P\{x=1\} \equiv P\{x\}$$

  Mientras que $\overline{P_x}$ representa la probabilidad de que la señal x sea igual a 0. Se consideran equivalentes estas representaciones:

  $$\overline{P_x} \equiv P\{x=0\} \equiv P\{\neg x\}$$

  Al tratar señales binarias, la relación entre éstas es:

  $$P_x = 1 - \overline{P_x} \tag{2.1}$$

- *Proceso estrictamente estacionario*, es un proceso en el que sus propiedades estadísticas no varían con el tiempo. Para el caso de los modelos del circuito, si es un circuito combinacional y sus entradas siguen un proceso estrictamente estacionario, entonces, sus señales internas también siguen un proceso estacionario en sentido estricto. Y por tanto:

$$P_x^{t0} = P_x^{t1} = P_x \tag{2.2}$$

Donde P_x^{t0} y P_x^{t1} son la probabilidades de la señal x en los tiempos t_0 y t_1 respectivamente: $P_x^{t0} \equiv P\{x(t_0)\}$.

Para simplificar la notación, el superíndice de una variable se referirá al tiempo. Por tanto, x^t denotará $x(t)$, es decir, el valor de la señal x en el tiempo t.

$$x^t \equiv x(t)$$

- *Probabilidad de transición*, $P(x_{i\to j})$, es la probabilidad de que la señal x experimente una transición de i a j ($i, j \in \{0, 1\}$), de un ciclo a otro[18]. Hay dos transiciones que conllevan cambio de valor: $x_{0\to1}$, $x_{1\to0}$; y otras dos que implican una permanencia en el estado previo $x_{0\to0}$, $x_{1\to1}$. Matemáticamente la probabilidad de transición se expresa:

$$P(x_{i\to j}) \equiv P\{(x^t = i) \wedge (x^{t+T} = j)\} \tag{2.3}$$

Donde $\wedge$ es el operador AND (Y-lógico), lo que implica que ambos eventos $x^t=i$ y $x^{t+T} = j$, se tienen que cumplir. T es el ciclo mínimo del circuito, que en circuitos síncronos corresponde con el ciclo de reloj.

Cada señal experimenta el mismo número de transiciones $x_{0\to1}$ que $x_{1\to0}$, ó a lo más habrá una transición de diferencia. Estas probabilidades de transición en procesos estrictamente estacionarios son iguales:

$$P(x_{0\to1}) = P(x_{1\to0}) \tag{2.4}$$

- *Actividad de conmutación* o *actividad de señal*, a_x, es la probabilidad de que la señal x experimente una transición que implique un cambio de valor.

$$a_x = P(x_{0\to1}) + P(x_{1\to0}) = 2 \cdot P(x_{0\to1}) \tag{2.5}$$

- *Inactividad de señal,* $\overline{a_x}$, es la probabilidad de que la señal x no experimente cambio de valor de un ciclo a otro.

$$\overline{a_x} = P(x_{0\to0}) + P(x_{1\to1}) = 1 - a_x \tag{2.6}$$

- *Eventos independientes* son aquellos en los que la ocurrencia de uno no tiene influencia en el resto. La probabilidad de que ocurran todo un conjunto de eventos independientes es igual al producto de probabilidades de dichos eventos tomados individualmente. Así, si A y B son independientes:

$$P\{A \wedge B\} = P_A \cdot P_B$$

Que generalizando, si $A_1, A_2, ..., A_n$, son n eventos independientes:

$$P\left\{\bigcap_{i=0}^{n} A_i\right\}_{indep} = \prod_{i=0}^{n} P_{A_i} \tag{2.7}$$

- *Eventos mutuamente excluyentes*, también llamados *eventos disjuntos*, son aquellos que la ocurrencia de uno de ellos impide la ocurrencia del resto. Es decir, no tienen ningún elemento en común. La intersección de eventos disjuntos es el conjunto vacío. Este tipo de eventos tiene una propiedad muy interesante para el cálculo de probabilidades: La probabilidad de que ocurran uno o más eventos disjuntos es igual a la suma de las probabilidades de dichos eventos tomados individualmente. Es decir, si A y B son disjuntos:

$$P\{A \vee B\} = P_A + P_B$$

Donde $\vee$ es el operador OR-lógico. Y generalizando para n eventos $A_1, A_2, ..., A_n$ disjuntos:

$$P\left\{\bigcup_{i=0}^{n} A_i\right\}_{disj} = \sum_{i=0}^{n} P_{A_i} \tag{2.8}$$

Y por la propia definición de eventos disjuntos:

[18] Se considerará que existe un ciclo mínimo (T) que determinará el tiempo mínimo entre dos transiciones. No se están considerando transiciones espurias.

$$P\left\{\bigcap_{i=0}^{n} A_i\right\}_{disj} = 0 \tag{2.9}$$

- *Señales disjuntas*, de manera similar a los eventos disjuntos, un conjunto de señales digitales son mutuamente disjuntas si como máximo sólo una señal puede ser verdadera en un tiempo dado [2]. Tratándose de lógica directa (no negada), en un momento dado sólo podrá haber una que valga un '1'-lógico. En caso de lógica negada, no podrá haber más de una señal que valga '0' en el mismo instante de tiempo. Si no se especifica lo contrario, se hablará de lógica directa.
- Las relaciones entre probabilidad de señal y probabilidad de transición son:

$$P_x = P(x_{1\to 0}) + P(x_{1\to 1}) = P(x_{0\to 1}) + P(x_{1\to 1}) = \tfrac{1}{2}\, a_x + P(x_{1\to 1}) \tag{2.10}$$

$$\overline{P_x} = P(x_{0\to 0}) + P(x_{0\to 1}) = P(x_{0\to 0}) + P(x_{1\to 0}) = \tfrac{1}{2}\, a_x + P(x_{0\to 0}) \tag{2.11}$$

  Que se demuestra:

$$\begin{aligned} P_x = P\{x^0\} &= P\left\{(x^0) \wedge \{\neg x^T \vee x^T\}\right\} = \\ &= P\left\{\{x^0 \wedge \neg x^T\} \vee \{x^0 \wedge x^T\}\right\} = \\ &= P\{x^0 \wedge \neg x^T\} + P\{x^0 \wedge x^T\} = \\ &= P(x_{1\to 0}) + P(x_{1\to 1}) \end{aligned}$$

  Pues $\{\neg x^T \vee x^T\}$ es un suceso cierto. Y $\{x^0 \wedge \neg x^T\}$ y $\{x^0 \wedge x^T\}$ son eventos disjuntos.

  El resto de relaciones se demuestra de manera análoga.

- *Probabilidad condicionada* del suceso *A* respecto de *B*, es la probabilidad de que ocurra *A* sabiendo que ha ocurrido *B*.

$$P\{A \mid B\} = \frac{P\{A \wedge B\}}{P\{B\}} \tag{2.12}$$

  Si *A* y *B* son independientes:

$$P\{A \mid B\}_{indep} = P_A$$

2.2.2 Diagramas de decisión binaria

En este apartado se explican algunas propiedades y el funcionamiento de los BDD. Para simplificar la notación, en esta tesis se ha llamado BDD a lo que más específicamente es un ROBDD (BDD ordenado y reducido).

Un BDD representa a una función booleana como un grafo acíclico orientado. Los BDD ofrecen grandes ventajas para la representación de funciones booleanas [16], [17], [128]:

- Forman una representación canónica de la función para un mismo orden de variables, por tanto facilitan la comprobación de propiedades funcionales y de equivalencias.
- Las operaciones sobre los BDD pueden ser implementadas por algoritmos con complejidad polinómica respecto a su tamaño. Como resultado, la manipulación de los BDD tiene dos ventajas:
  - Siempre que el BDD tenga un tamaño moderado, la computación es tratable.
  - Aunque el tamaño pueda crecer en cada operación sucesiva, cualquier operación tiene unas prestaciones razonables para el peor caso.
- La función resultante de un BDD está representada por un conjunto de cubos que forman una cubierta disjunta de la función. Ésta es una propiedad muy interesante para el cálculo de la probabilidad de una función lógica.

Por estas razones, los BDD son ampliamente empleados en muchas tareas de diseño como síntesis lógica, verificación y test, así como para el cálculo de la probabilidad y actividad de las señales de un circuito.

La obtención del BDD de una función lógica se hace mediante la reducción del árbol de decisión mostrado en la figura 2-8. En éste, las ramas de trazo continuo indican que la variable toma valor 1 y se denominarán **ramas uno**; mientras que las ramas de trazo punteado indica que la variable toma el valor 0 y éstas se llamarán **ramas cero**[19]. Al primer nodo del árbol de decisión y del BDD resultante se le llama **nodo raíz**.

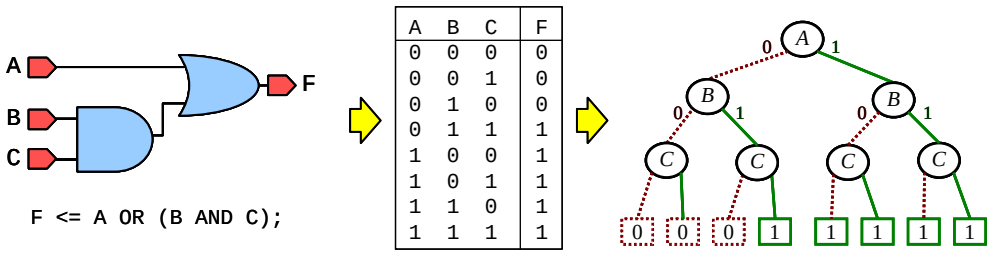


Figura 2-8: Circuito en puertas lógicas, tabla de verdad y árbol de decisión

El proceso de reducción que pasa del árbol de decisión al BDD se basa en juntar los nodos duplicados y eliminar los nodos con alternativas iguales. Este proceso se encuentra detallado en las referencias [16], [17]. El BDD resultante para este ejemplo se muestra en la figura 2-9, en donde además se indica cómo se obtienen las ecuaciones para la función y la función negada. Cada camino que termina en 1 se suma a la función, mientras que cada camino que termina en 0 se suma a la función negada. A estos nodos cero y uno se les llamará nodos **terminales** o **finales**.

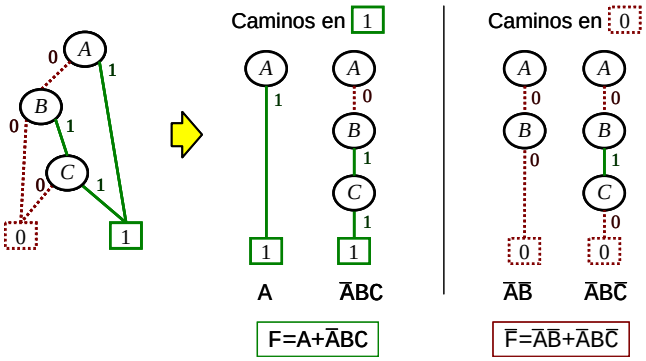


Figura 2-9: BDD para la función lógica del ejemplo 2-8 y obtención de las ecuaciones para la función lógica y su negada

El **recorrido de un BDD** se realiza partiendo de su nodo raíz. Sin embargo, para recorrer un BDD, cada camino no se empieza desde el nodo raíz, sino que se utiliza un algoritmo con llamadas recurrentes a la función que lo atraviesa. En este algoritmo se va guardando el camino transitado desde el nodo raíz y se va actualizando al entrar y salir de cada nodo. Este algoritmo es bien conocido en la literatura de grafos, en el cual al llegar a cada nodo se vuelve a llamar a la función para cada uno de sus hijos. El anexo AI.1 muestra la estructura del código que implementa este algoritmo.

Así, el recorrido del BDD de la figura 2-9 siguiendo el algoritmo descrito se muestra en la figura 2-10. En ella se muestra la secuencia de pasos en el recorrido, estos pasos se han marcado con un número insertado en un hexágono bajo el BDD correspondiente. En el primer paso se ha recorrido el camino que va por las ramas cero de *A* y *B*. En el segundo paso, después de llegar al nodo final cero se vuelve al nodo *B*. En el tercer paso, se toma la rama uno del nodo *B*, llegando

[19] En los dibujos se distinguirán por estar en trazo continuo las ramas uno, mientras que las ramas cero estarán en trazo discontinuo. Para ayudar a recordar, en algunos dibujos se incluirán un cero o un uno al lado de la rama.

al nodo *C*. En el cuarto paso se toma la rama cero de *C* llegando al nodo final cero, con lo que se obtiene otro camino terminado en cero. En el quinto paso, se vuelve al nodo *C* y se toma su rama uno, llegando al nodo final uno, obteniéndose el camino terminado en uno. Por último, en el paso sexto, se recorre de vuelta el camino hasta el nodo raíz (*A*), y se toma su rama uno, que llega directamente al nodo final uno, resultando en el último camino del BDD.

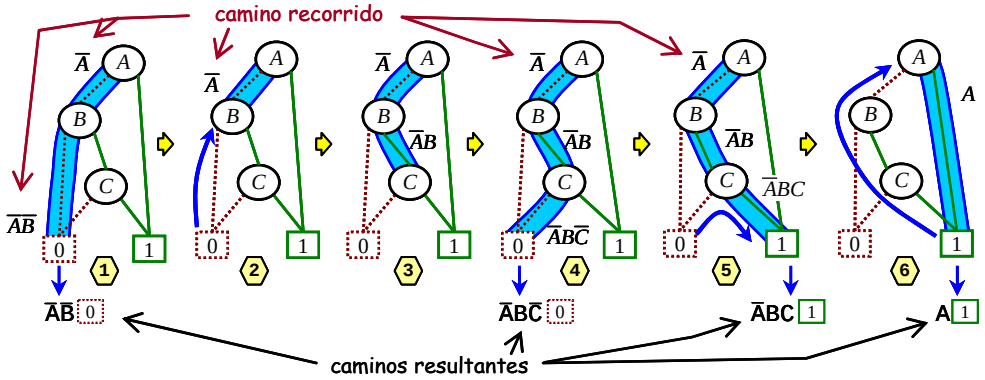


Figura 2-10: Recorrido del BDD de la figura 2-9 y obtención de sus caminos

No obstante, existe un método más eficiente que utiliza un enfoque dinámico. Este método se ha explicado en el apartado AI.2 del anexo (figura AI-2).

El **orden de las variables de un BDD** es importante a la hora de conseguir un tamaño reducido. La figura 2-11 muestra los BDD resultantes de variar el orden de las variables del BDD del ejemplo de la figura 2-9. Nótese cómo el BDD del medio tiene un nodo más. Para funciones con mayor número de entradas el orden de las variables es un aspecto fundamental, ya que la variación puede ser más críticos. En algunos casos, la elección de un determinado orden u otro puede hacer que el número de nodos aumente lineal o exponencialmente con el número de entradas [17].

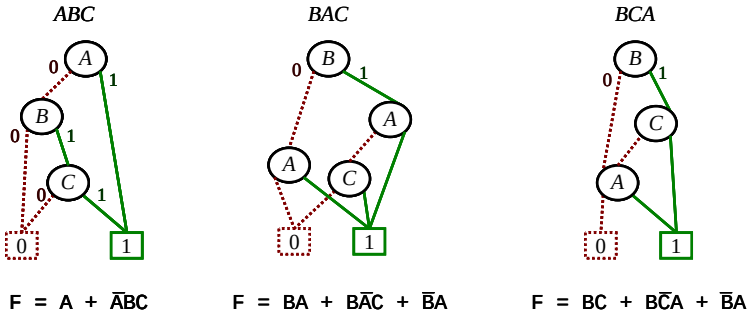


Figura 2-11: BDD y ecuaciones resultantes con distintos órdenes de variable para el ejemplo de la figura 2-8

Los diagramas de decisión binaria representan gráficamente la funcionalidad de una función lógica. Considérese la función $f(x_1,..., x_i,..., x_n)$ de variables x_i independientes. La función f puede representarse mediante la *expansión de Shannon* de la siguiente manera:

$$f = x_i \cdot f(x_1,..., x_{i-1}, 1, x_{i+1},..., x_n) + \bar{x}_i \cdot f(x_1,..., x_{i-1}, 0, x_{i+1},..., x_n) \quad (2.13)$$

Definiéndose los cofactores de f respecto a x_i y respecto a $\bar{x}_i$, respectivamente:

$$f_{x_i} = f(x_1,..., x_{i-1}, 1, x_{i+1},..., x_n) \quad (2.14)$$

y

$$f_{\bar{x}_i} = f(x_1,..., x_{i-1}, 0, x_{i+1},..., x_n)$$

Las funciones f_{x_i} y $f_{\bar{x}_i}$ se obtienen reemplazando la variable x_i con un 1 ó con un 0 lógico respectivamente.

Esto se puede hacer recursivamente para todas las variables (x_i). El orden que se tome determinará el orden del BDD. La figura 2-12 muestra la expansión de Shannon para el ejemplo

de la figura 2-9. En donde se ha ido realizando la expansión en tres pasos siguiendo el orden: *A*, *B* y *C*. El último árbol se convierte en un BDD siguiendo los pasos de reducción [16], [17].

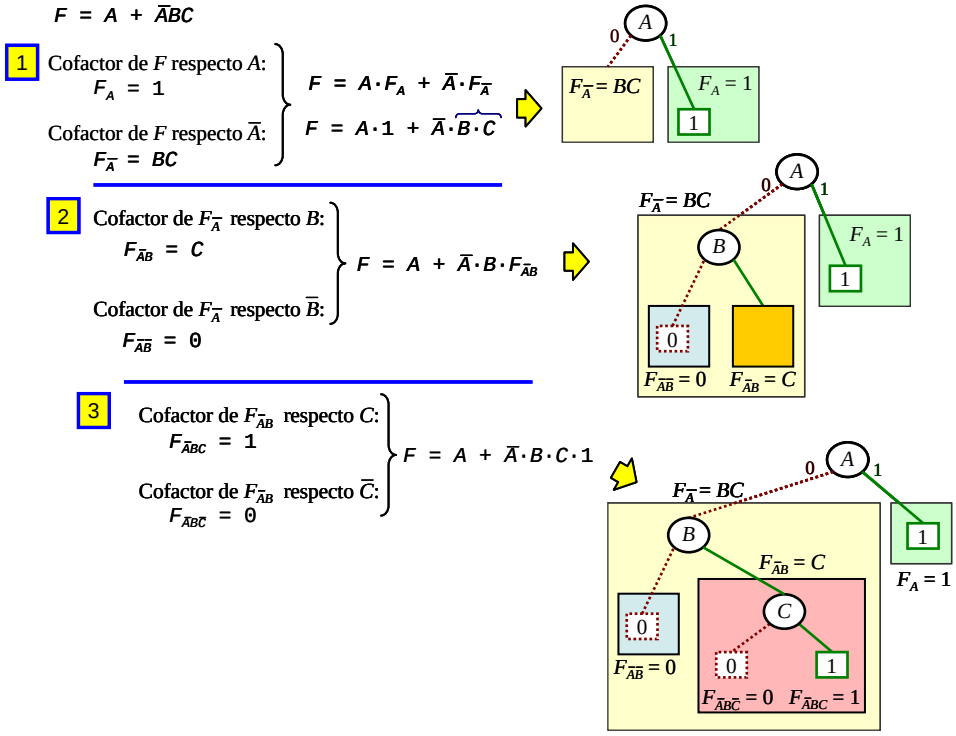


Figura 2-12: Expansión de Shannon para el ejemplo de la figura 2-9

Debido a que el evento $\{x_i \wedge \neg x_i\}$ no puede suceder nunca, esto es $\{x_i \wedge \neg x_i\} = \varnothing$, la expansión de Shannon se descompone en sumandos mutuamente disjuntos, entonces sus probabilidades se pueden separar. Pasando de:

$$P_f = P\{(x_i \wedge f_{x_i}) \vee (\neg x_i \wedge f_{\overline{x_i}})\} \tag{2.15}$$

a:

$$P_f = P\{x_i \wedge f_{x_i}\} + P\{\neg x_i \wedge f_{\overline{x_i}}\} \tag{2.16}$$

Y como los cofactores de x_i no dependen de x_i, y todas las variables se consideran independientes, también se pueden separar:

$$P_f = P_{x_i} \cdot P(f_{x_i}) + \overline{P_{x_i}} \cdot P(f_{\overline{x_i}}) \tag{2.17}$$

Esta ecuación muestra cómo se puede emplear el BDD para propagar la probabilidad de una señal, evaluando P_f. Los dos nodos que descienden de *f* en el BDD se corresponden con los cofactores de *f*. Y de la misma manera, la probabilidad de los cofactores se puede expresar en términos de sus descendientes. Expresando recursivamente todos los cofactores en términos de sus descendientes, de la ecuación 2.17 se obtiene una suma de productos (SOP) de probabilidades de las entradas, que representa una cubierta disjunta del conjunto que activa la función (*onset*). Ésta es una de las propiedades que hace tan interesante el uso de BDD.

En el ejemplo de la figura 2-9, para obtener de la ecuación de probabilidades, el hecho de que cada elemento de la función sea mutuamente disjunto con el resto, permite que se puedan separar las probabilidades:

$$P_F = P(A + \overline{A}BC) = P(A) + P(\overline{A}BC) \tag{2.18}$$

Y si se consideran entradas mutuamente independientes, se puede separar el segundo sumando, y quedaría:

$$P_F = P_A + \overline{P}_A \cdot P_B \cdot P_C \tag{2.19}$$

2.2.3 Modelo probabilístico

Una vez que se han definido los conceptos probabilísticos, su terminología asociada y se han esbozado las nociones relativas a los BDD, en esta sección se sentarán las bases de los modelos probabilísticos para la estimación de la probabilidad y actividad. Además, en esta sección se presentarán las dificultades más importantes de la elaboración del modelo.

El modelo probabilístico de un circuito calcula las probabilidades de que cada señal tome un valor y sufra una transición. Según las consideraciones que se hagan variarán la complejidad y exactitud del modelo.

Para el cálculo de las probabilidades se recurre a la ecuación de la función lógica de la señal pero, en vez de calcular la salida en función de los valores binarios de las entradas, se calcula la probabilidad de que la salida tenga un valor concreto en función de las probabilidades de las entradas. Como los circuitos lógicos implementan lógica digital, la traducción de las funciones lógicas a probabilidades de eventos es inmediata. Así, como muestra la figura 2-13, la probabilidad de la señal de salida de una puerta AND se traduce en la probabilidad de que en ambas entradas de la puerta haya un uno, esto es: $P\{A \wedge B\}$. Para una puerta OR, se calcula la probabilidad de que una de las entradas, o las dos, sean uno: $P\{A \vee B\}$.

Para simplificar el cálculo de la probabilidad de estos eventos se hace uso de las propiedades de los eventos independientes y de los eventos excluyentes (ecuaciones 2.7 y 2.8).

Así, si se considera que las señales de entrada (A y B) de la puerta AND son independientes, su probabilidad quedaría $P_{AND}=P_A \cdot P_B$. Sin embargo, como se aprecia a la derecha de la figura 2-13, si no se cumple la condición de independencia en las entradas, el resultado puede ser totalmente distinto. En este caso se han incluido casos extremos, si las entradas se corresponden con eventos exclusivos, la probabilidad de la salida será cero. Mientras que si las entradas se corresponden con el mismo evento, la probabilidad de la salida es la probabilidad de las entradas. Ya que, evidentemente, las entradas tendrán la misma probabilidad por tratarse de la misma señal.

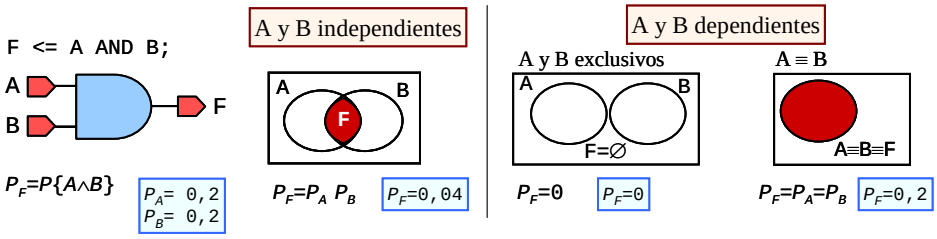


Figura 2-13: Probabilidad de la salida de la puerta AND para diversas situaciones de correlaciones en las entradas

Algo parecido ocurre con la puerta OR, en la cual, para separar la expresión del evento $\{A \vee B\}$ se debe expresar como una disyunción de eventos excluyentes (o disjuntos). Normalmente no hay una única disyunción de eventos excluyentes, cada fila de la tabla de verdad es un evento disjunto. Por ejemplo, bien se puede tomar la forma de suma canónica de productos (figura 2-6); escoger la mostrada en la figura 2-14; o cualquier otra posibilidad. Una vez que está expresada en términos disjuntos se puede separar en suma de probabilidades de eventos disjuntos y, si los eventos dentro de cada sumando de probabilidad son independientes se pueden volver a separar, esta vez en forma de producto de probabilidades (figura 2-14). Nótese otra vez el resultado tan distinto que se obtiene si no se cumple la condición de independencia entre las entradas.

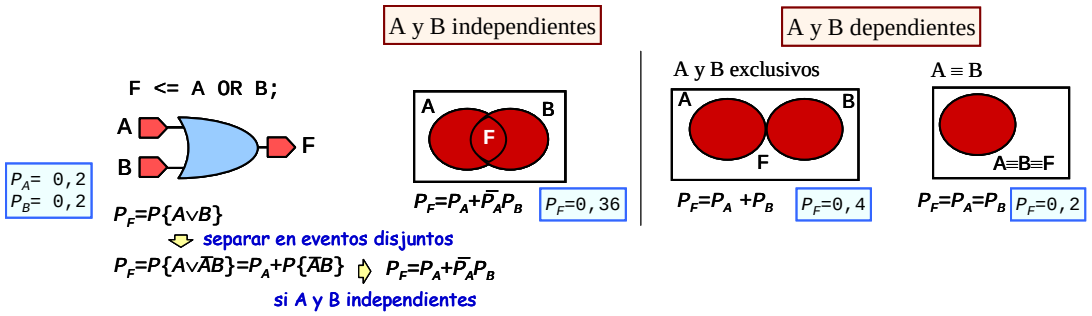


Figura 2-14: Probabilidad de la salida de la puerta OR para diversas situaciones de correlaciones en las entradas

Por tanto, **la capacidad del modelo probabilístico para considerar las dependencias entre las señales determina su fiabilidad y precisión**, y a su vez, **su complejidad y el tiempo empleado en la estimación**.

En los siguientes apartados se expondrán ciertas **características** de los circuitos que se pueden considerar a la hora de realizar un modelo probabilístico, pues afectan a la actividad y a las interdependencias entre señales. No todas las características tienen que ser consideradas, dependerá del compromiso entre la complejidad y fiabilidad del modelo. Estas características son:

- Dependencias estructurales
- Dependencias temporales
- Dependencias espaciales en las entradas
- Retardos en las puertas
- Elementos de memoria

2.2.3.1 Dependencias estructurales

Una característica del circuito que afecta de manera muy importante a la exactitud de los resultados son las dependencias estructurales.

En un circuito existen dependencias estructurales cuando existen caminos de señal que reconvergen en un punto, lo que también es conocido como *fan-out* reconvergente. La figura 2-15 muestra un ejemplo de un circuito en el que la señal *A* influye en las señales *D* y *E*, por tanto estas señales no son independientes. Como consecuencia, no se puede calcular la probabilidad de *F* directamente en función de las probabilidades de *D* y *E*. Es decir $P_F = P\{D \wedge E\} \neq P_D \cdot P_E$.

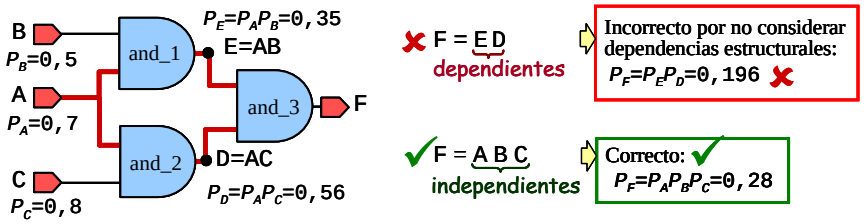


Figura 2-15: Influencia de las dependencias estructurales en el cálculo de las probabilidades

Para evitar el efecto de las dependencias estructurales, se debe ir hacia atrás en el circuito en busca de señales independientes. Para el caso de la figura 2-15, suponiendo los puertos de entrada independientes, se pone la señal *F* en función de *A*, *B* y *C*.

La ecuación 2.20 muestra cómo se llega a la ecuación de probabilidades con señales independientes. En ella se puede observar cómo se simplifica el evento repetido *A*, que es quien

provoca la dependencia entre *D* y *E*, y que si se calculase en función de estas últimas su probabilidad se contabilizaría dos veces.

$$P_F = P\{D \wedge E\} = P\{(C \wedge A) \wedge (A \wedge B)\} =$$
$$P\{C \wedge A \wedge A \wedge B\} = P\{C \wedge A \wedge B\} = P_A \cdot P_B \cdot P_C \quad (2.20)$$

Los circuitos con muchas dependencias estructurales complican la elaboración del modelo debido a que ya no se pueden propagar las probabilidades y actividades puerta a puerta, sino que hay que ir en busca de las fuentes independientes de cada señal. Esto hace que para circuitos grandes se disparen los recursos computacionales y de memoria requeridos.

2.2.3.2 Dependencias temporales

Las dependencias temporales tienen una influencia fundamental en el cálculo de la actividad y por tanto es importante tenerlas en cuenta en el modelo probabilístico.

Cuando no hay dependencias temporales, el valor de una señal es independiente de su valor en los ciclos precedentes. En esta situación es correcto calcular la actividad de conmutación partiendo únicamente de las probabilidades de señal.

Así, en condiciones de independencia temporal, la probabilidad de que sucedan dos eventos de una misma señal en tiempos diferentes es igual al producto de las probabilidades de cada uno de los eventos. Así, si se tiene que

$$P(x_{0\to 1}) = P\{\neg x^0 \wedge x^T\}$$

Donde *T* es el ciclo mínimo del circuito.

Y si se considera que x^0 y x^T son independientes, por la ecuación 2.7, se pueden calcular las probabilidades de sus términos separadamente:

$$P(x_{0\to 1}) = P\{\neg x^0\} + P\{x^T\} = \overline{P_x^0}\, P_x^T$$

Considerando que las señales se comportan como un proceso estrictamente estacionario (ecuación 2.2), se quitan los tiempos de las probabilidades:

$$P(x_{0\to 1}) = \overline{P_x}\, P_x$$

Y sustituyendo en la ecuación de la actividad (ecuación 2.5), queda:

$$a_x^{it} = 2\, P_x\, \overline{P_x} = 2P_x\,(1 - P_x) = 2\, P_x - 2\,(P_x)^2 \quad (2.21)$$

Donde el superíndice *it* de la actividad recuerda que se ha calculado considerando independencia temporal.

Como se puede apreciar, con esta suposición, el cálculo de la actividad de conmutación es directo a partir de la probabilidad de señal, simplificando de manera extraordinaria los cálculos de la actividad de conmutación.

Sin embargo, no siempre es válido considerar que en las señales no existe correlación temporal, ya que el valor de una señal suele depender de su valor anterior. En la figura 2-16 se muestra cómo distintas señales con la misma probabilidad de señal P_x, pueden tener una actividad de conmutación muy diferente.

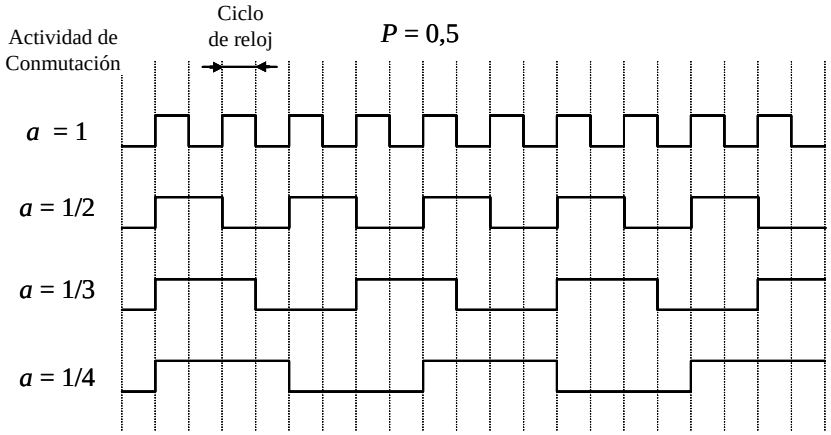


Figura 2-16: Formas de ondas de señales con la misma probabilidad (P=0,5) pero diferente actividad de conmutación

Consecuentemente, no tener en cuenta las correlaciones temporales de las señales puede dar lugar a imprecisiones en el cálculo de la actividad de conmutación, *a*. No obstante, la probabilidad de señal y la actividad de conmutación están relacionadas por la expresión:

$$½\ a_x \le P_x \le 1 - ½\ a_x \tag{2.22}$$

Que se puede reescribir como:

$$a_x \le 1 - 2 \cdot |P_x - 0{,}5| \tag{2.23}$$

Estas ecuaciones se obtienen a partir de las ecuaciones 2.10 y 2.11:

$$P_x = ½\ a_x + P(x_{1\to1})$$

$$\overline{P_x} = 1 - P_x = ½\ a_x + P(x_{0\to0})$$

Como $P(x_{1\to1})$ y $P(x_{0\to0})$ sólo toman valores positivos entre 0 y 1. Fácilmente se transforman en las desigualdades 2.22 y 2.23.

La representación de la desigualdad 2.22 se muestra en la figura 2-17. El área dentro del triángulo indica los posibles valores de la actividad de conmutación dada una probabilidad de señal. Véase cómo la suposición de que las señales no tienen correlación temporal (la parábola que representa la ecuación 2.21) es tan sólo un caso particular de las múltiples opciones.

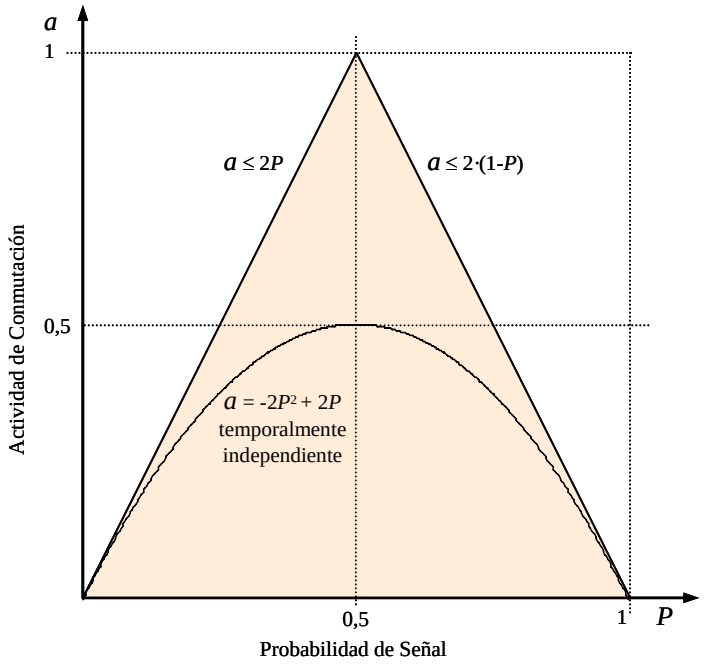


Figura 2-17: Relación entre actividad de conmutación y probabilidad de señal

Consecuentemente, para el cálculo de la actividad de conmutación considerando las dependencias temporales hay que calcular específicamente la ecuación de actividad. Como se puede observar de la figura 2-5, la consideración de las dependencias temporales complica mucho su cálculo.

2.2.3.3 Dependencias espaciales en las entradas

Las señales de entrada de un circuito pueden ser dependientes entre sí, esta circunstancia complica aún más la situación descrita en el apartado 2.2.3.1 referente a las correlaciones estructurales, pues ahora las correlaciones espaciales a las entradas no surgen por la estructura del circuito, que es visible, sino que aparecen debidas a unas condiciones externas a lo que se está estudiando. Por tanto, para tener en cuenta estas correlaciones, éstas se deben establecer y realizar un modelo del circuito que les de soporte.

2.2.3.4 Retardos en las puertas

Como ya se ha visto en la sección 1.3.3 las transiciones espurias hacen que una señal cambie inútilmente de valor antes de llegar a su estado final, provocando que la señal pueda tener más de una transición durante el ciclo de reloj. Si se construye un modelo que considere estas transiciones espurias, la actividad de señal no puede ser definida como una probabilidad debido a que las probabilidades nunca pueden ser mayores que la unidad. Por tanto habría que definirla como el número de transiciones esperado en un ciclo de reloj.

Aquellos modelos que consideran estas transiciones espurias se denominan modelos de **retardo real** (*real-delay*) en oposición a los modelos **sin retardo** (*zero-delay*).

La consideración de las transiciones espurias en la estimación de la actividad, aunque siempre costosa computacionalmente, es más factible en el nivel de puertas que en el nivel RT, ya que en este último nivel aún no está definida la disposición de la lógica, pudiendo una misma funcionalidad implementarse con muy distinta tendencia a la generación de espurios (ver figura 1-7). Por tanto, desde el nivel RT, estas transiciones dependen de las optimizaciones que se hagan durante la síntesis lógica.

Para el estimación de actividad con modelos de retardo real se puede utilizar el concepto de *función booleana en tiempos* (TBF: *Timed Boolean Function*) [73] y su representación en diagramas de decisión binaria: TBF-BDD.

2.2.3.5 Elementos de memoria

La elaboración del modelo probabilístico de circuitos con elementos de memoria con realimentación entraña gran dificultad, ya que los nodos del circuito dependen no sólo de las señales de entrada en el momento actual, sino también del estado interno de los elementos de memoria. Esta dependencia en los estados internos hace que, en el caso más general, el valor de los nodos del circuito dependa del estado inicial del circuito más toda la secuencia de entrada desde ese estado inicial hasta el momento presente. Este hecho implica recurrencia en la computación y la necesidad de simplificación en el análisis.

Para el caso de circuitos secuenciales sin realimentación la resolución es más sencilla debido a que la dependencia temporal máxima se limita al valor de la profundidad secuencial del circuito.

Una manera de abordar la elaboración del modelo probabilístico para los circuitos secuenciales con realimentación es extraer las probabilidades de transición entre los estados internos y analizarlas mediante el gráfico de transición de estados (STG[20]).

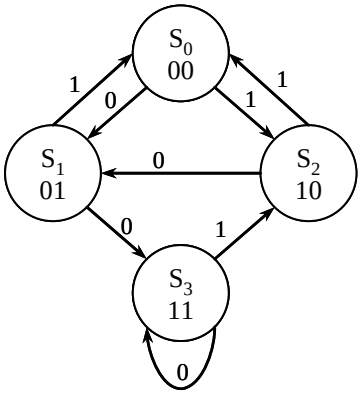


Figura 2-18: Gráfico de transición de estados de un circuito ejemplo

Dados N biestables, resultan 2^N ecuaciones a partir del gráfico de transición de estados. Este sistema de ecuaciones se conoce como las ecuaciones de Chapman-Kolmogorov para un proceso de Markov de transiciones y tiempos discretos [133]. Del ejemplo de la figura 2-18 resultan las siguientes ecuaciones:

$$P(S_0) = P_i \cdot P(S_2) + P_i \cdot P(S_1)$$
$$P(S_1) = \overline{P_i} \cdot P(S_0) + \overline{P_i} \cdot P(S_2)$$
$$P(S_2) = P_i \cdot P(S_0) + P_i \cdot P(S_3)$$
$$P(S_0) + P(S_1) + P(S_2) + P(S_3) = 1$$

Donde P_i es la probabilidad de la señal de entrada y $P(S_i)$ es la probabilidad del estado S_i. Conocidas las probabilidades de las entradas el sistema es lineal. Pero para circuitos grandes el método no es aplicable debido al número de ecuaciones que se generan (2^N).

Hasta aquí se han resumido los fundamentos de la elaboración del modelo y las dificultades más importantes con las que se enfrentan: dependencias estructurales, dependencias temporales, dependencias espaciales en las entradas, retardos en las puertas y elementos de memoria. En el apartado siguiente se exponen las propuestas para la elaboración del modelo probabilístico y cómo resuelven dichas dificultades.

2.3 Trabajo previo en la estimación probabilística

En este apartado se expondrán los trabajos más significativos que se han propuesto en el campo de la estimación probabilística. Los primeros modelos probabilísticos eran más básicos. Estos modelos se fueron completando y aumentando en complejidad a medida que la comunidad científica realizaba aportaciones.

Existe una gran variedad de propuestas realizadas que abordan los distintos problemas con que se enfrentan los modelos probabilísticos (dependencias espaciotemporales, elementos de memoria, retardos, etc.) vistos en el apartado anterior (§2.2.3). La exposición de los trabajos previos se ha organizado según las características del modelo propuesto. Sin embargo, no siempre se pueden clasificar las propuestas en un único grupo, ya que es habitual que las propuestas consideren más de un aspecto. Por tanto, estas propuestas, o bien aparecerán en más de un grupo, o bien se incluirán en el grupo en el que la aportación sea más significativa o novedosa.

Las propuestas se han clasificado en:

[20] Del inglés, *State Transition Graph*

- Modelos que consideran dependencias estructurales (§2.3.1)
- Modelos que consideran correlaciones temporales (§2.3.2)
- Modelos que consideran correlaciones espaciales en las entradas (§2.3.3)
- Modelos que consideran los retardos en las puertas (§2.3.4)
- Modelos que consideran los elementos de memoria (§2.3.5)
- Modelos que consideran otros niveles de abstracción (§2.3.6)
- Modelos a mitad de camino entre los probabilísticos y simulativos (§2.3.7)

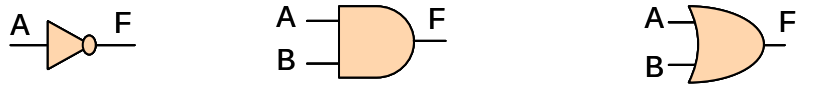
El último grupo (§2.3.7) no contiene modelos que incluyan una característica diferente a los demás, sino que es un tipo de modelo a mitad de camino entre los modelos probabilísticos y los simulativos.

2.3.1 Primeros modelos y modelos que consideran dependencias estructurales

Los primeros modelos propuestos son los más simples y normalmente sólo consideran las dependencias estructurales.

Las técnicas probabilísticas surgieron del análisis de la probabilidad de señal para el análisis de circuitos con fallos. La probabilidad de señal constituye un paso imprescindible para el estudio probabilístico de la actividad de conmutación y, en condiciones de independencia temporal de las señales, la actividad se puede deducir directamente a partir de la probabilidad (§2.2.3.2, ecuación 2.21).

Uno de los primeros trabajos relativos al análisis probabilístico de fallos fue realizado por **Parker** [103]. En este trabajo, para cada señal x de un circuito combinacional se obtiene su probabilidad, P_x . Las señales se tratan como variables aleatorias. Para ello se asocia cada puerta lógica con una expresión de probabilidad. En la figura 2-19 se muestran las expresiones de probabilidad propuestas para varias puertas lógicas.



$P_F = 1 - P_A$ $\qquad$ $P_F = P_A P_B$ $\qquad$ $P_F = P_A + P_B - P_A P_B$

Figura 2-19: Ecuaciones de probabilidad para varias puertas lógicas

Para circuitos sin dependencias estructurales[21] (§2.2.3.1) se pueden emplear las fórmulas anteriores puerta por puerta, desde las entradas a las salidas, haciendo que la propagación de probabilidades sea rápida y sencilla. No obstante, los circuitos con dependencias estructurales generan correlaciones entre las señales internas del circuito que pueden hacer que las señales a las entradas de las puertas lógicas no sean independientes. Esta dependencia de las entradas provoca errores en el cálculo de la probabilidad (recuérdese la figura 2-15). Si se toman las señales independientes la complejidad del método puede aumentar excesivamente hasta hacerlo inviable para circuitos grandes.

Continuando con los estudios para testabilidad y por tanto, restringido a probabilidad de señal, **Seth** [121], [122], en la herramienta PREDICT, trata las dependencias estructurales realizando particiones del circuito en lo que llama superpuertas (*supergates*), de modo que cada superpuerta sólo contenga entradas mutuamente independientes. La figura 2-20 muestra el agrupamiento de puertas en superpuertas.

[21] Sin fan-out reconvergente

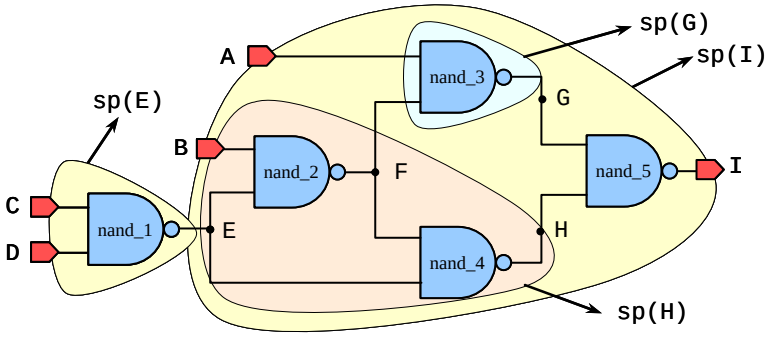


Figura 2-20: Superpuertas (sp) para señales de un circuito de ejemplo

Para circuitos grandes y con muchas reconvergencias las superpuertas abarcan gran parte del circuito e incluso pueden contener a todas las entradas primarias del circuito. Esto puede hacer que se dispare la utilización de recursos computacionales y de memoria. En un cálculo aproximado, se propone limitar la profundidad de la superpuerta a una distancia máxima determinada (ver figura 2-21).

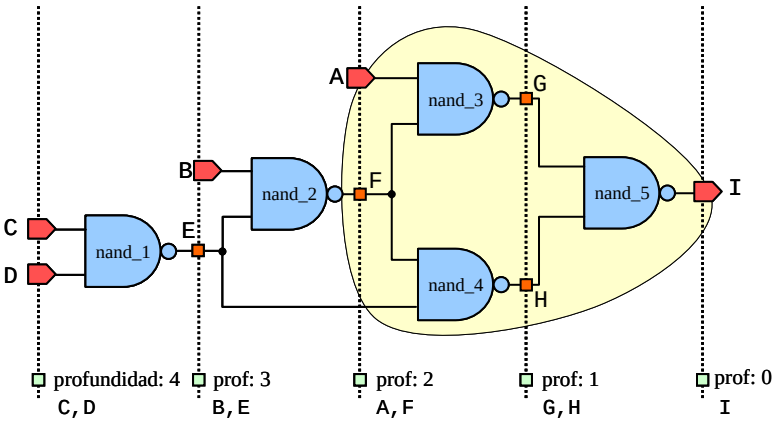


Figura 2-21: Profundidades respecto a la señal I, y su superpuerta con profundidad limitada a 2

El primer trabajo que relaciona las probabilidades de señal con la actividad de conmutación y el consumo dinámico fue llevado a cabo por **Cirit** [29], implementándolo en la herramienta LTIME. En el cálculo se asume independencia temporal, por lo que la actividad se calcula según la ecuación 2.21:

$$a_x^{it} = 2\,P_x\,\overline{P_x} = 2\,P_x\,(1-P_x)$$

Donde el superíndice *it* de la actividad recuerda que se ha calculado considerando independencia temporal.

Para evitar el excesivo tamaño de las zonas reconvergentes, **Agrawal** [2] propone métodos para buscar señales mutuamente disjuntas en circuitos combinacionales descritos en nivel de puertas. Las señales mutuamente disjuntas permiten realizar particiones del circuito, lo que facilita el cálculo de la probabilidad. El algoritmo propuesto está basado en análisis de contradicción y detecta, mediante un proceso similar a la simulación, aquellas señales que son disjuntas. De todas maneras, el algoritmo no es capaz de detectar todas las señales disjuntas ya que en tal caso se tendrían realizar análisis simbólicos que son más complejos y requieren mayores tiempos de cómputo y uso de memoria.

La figura 2-22 muestra cómo el conocimiento de que las señales *D* y *E* son disjuntas permite dividir el circuito y calcular la probabilidad de la señal *F* solamente en función de sus entradas, sin necesidad de buscar las fuentes independientes *A*, *B* y *C*. En la misma figura se ha incluido un ejemplo numérico para facilitar la apreciación de este fenómeno. Esta propuesta se restringe al cálculo de la probabilidad, donde el resultado es exacto. Sin embargo, para el cálculo de la actividad se introduce error (véase el apartado 3.4.3)

Nótese la diferencia con el circuito de la figura 2-15. En aquel circuito las señales *D* y *E* también dependían de una misma señal común, sin embargo la estructura del circuito no hacía que fuesen disjuntas. Como consecuencia, para el circuito de la figura 2-15, el cálculo de la probabilidad directamente a partir de las señales *D* y *E* produce resultados incorrectos.

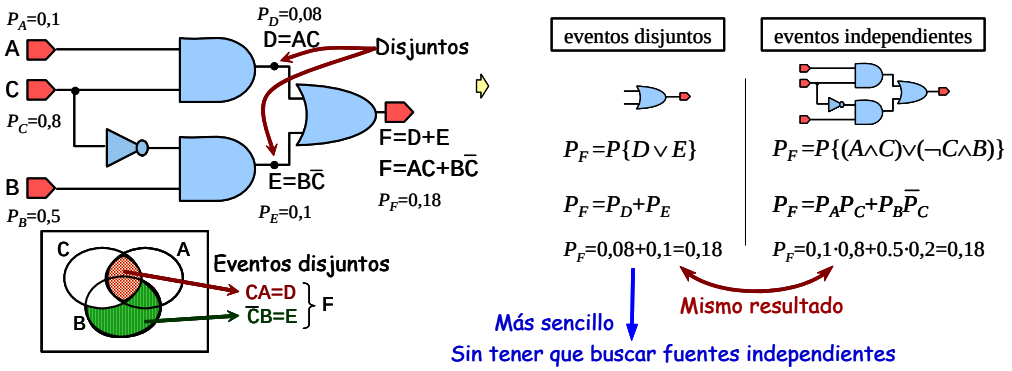


Figura 2-22: Señales disjuntas pueden simplificar el cálculo de la probabilidad

2.3.2 Modelos que consideran las correlaciones temporales

Najm [90] introduce las correlaciones temporales mediante lo que denomina densidad de transición. La densidad de transición es un concepto relacionado con la actividad y es utilizada para calcular el consumo (ecuación 1.3). La propuesta se implementa en la herramienta DENSIM. Najm propone un algoritmo para propagar las probabilidades y densidades de transición desde las entradas a lo largo de todo el circuito mediante el uso de *diagramas de decisión binaria* (BDD) [16]. Una explicación del funcionamiento y las propiedades de los BDD se encuentra en la sección 2.2.2.

Para la propagación de la densidad de transición, Najm recurre al concepto de *diferencia booleana* ($\partial f/\partial x_i$):

$$\frac{\partial f}{\partial x_i} = f\big|_{x_i=1} \oplus f\big|_{x_i=0} = f(x_i) \oplus f(\bar{x}_i) \tag{2.24}$$

En donde f es una función que depende de x_i; el símbolo $\oplus$ representa el operador OR-exclusivo (XOR). La diferencia booleana indica la condición que hace sensible la función f respecto a cambios en x_i. Nótese que $\partial f/\partial x_i$ es una función booleana que no depende de x_i, aunque puede depender del resto de entradas x_j. Por tanto $\partial f/\partial x_i$ y x_i son independientes. En el caso extremo en que $\partial f/\partial x_i = 1$, una transición de x_i provocará una transición en f.

Utilizando la densidad de transición y considerando que no hay correlaciones espaciales entre las entradas primarias x_i, (i = 1,..., n) de la función f, la densidad de transición, $D(f)$, se calcula mediante la fórmula:

$$D(f) = \sum_{i=1}^{n} P\left(\frac{\partial f}{\partial x_i}\right) \cdot D(x_i) \tag{2.25}$$

Como la diferencia booleana $\partial f/\partial x_i$ simboliza la sensibilidad de la función f respecto a las transiciones en x_i, su probabilidad, $P(\partial f/\partial x_i)$, representa la probabilidad de que una transición en x_i afecte a la salida f. Esta probabilidad multiplicada por la densidad de transición de x_i, $D(x_i)$, resulta en la contribución de la señal x_i a la densidad de transición de la salida f. Por tanto, la contribución a la densidad de transición debida a todas las entradas se obtiene sumando la contribución de todas las entradas de f. En la figura 2-23 se muestra el cálculo de la densidad de transición para algunas puertas.

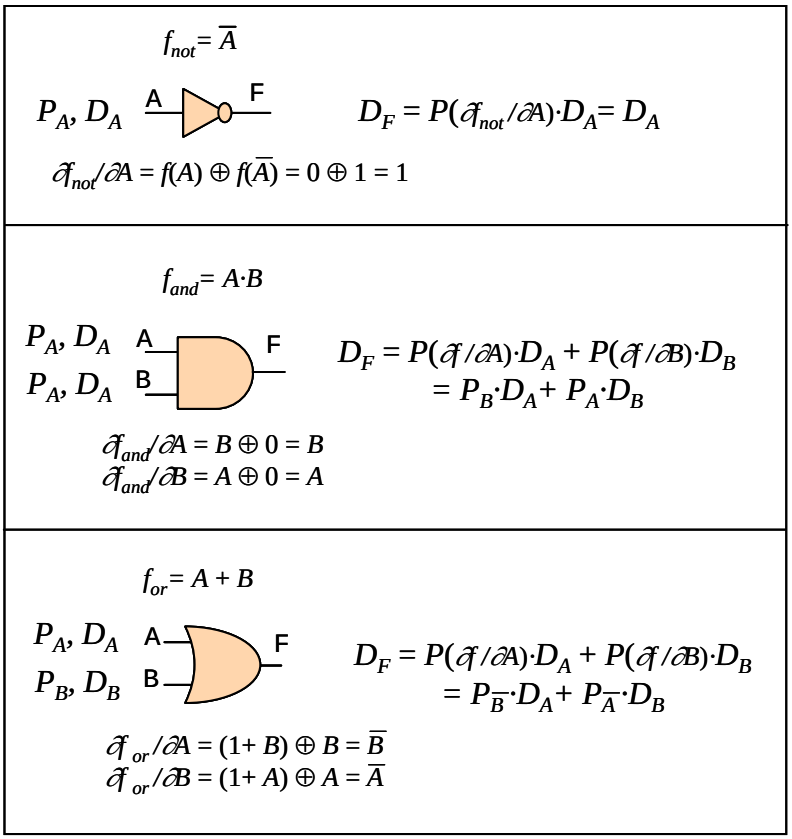
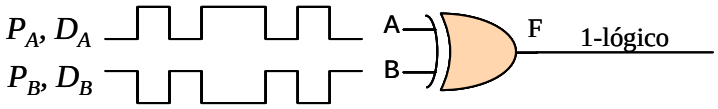


Figura 2-23: Propagación de densidades de transición a través de puertas básicas

Sin embargo, la ecuación 2.25 no considera el efecto de las posibles **transiciones simultáneas** de las entradas y esto puede llevar a un mal cálculo de la densidad de transición. Con una puerta XOR se puede apreciar claramente esta situación. En la figura 2-24 se muestra cómo la conmutación simultánea de entradas hace que no se traduzca en una conmutación en la salida, a pesar de que para cada entrada la diferencia booleana, $\partial f/\partial x_i$, de la salida respecto a la entrada es uno[22]. Mediante la ecuación 2.25, la densidad de transición de la salida (D_f) para la puerta XOR resulta en la suma de las densidades de transición de sus entradas. Pero ésta sólo sería válida en caso de que no hubiese transiciones simultáneas en las entradas, pues son estas transiciones simultáneas las que hacen que no haya transición a la salida.

$$f_{xor} = A \oplus B = \overline{A}\cdot B + A\cdot\overline{B}$$



$$\partial f_{xor}/\partial A = \overline{B} \oplus B = 1 \qquad D_F = P(\partial f/\partial A)\cdot D_A + P(\partial f/\partial B)\cdot D_B$$
$$\partial f_{xor}/\partial B = \overline{A} \oplus A = 1 \qquad = D_A + D_B$$

Figura 2-24: Sobreestimación de la densidad de transición para una puerta XOR

Cuando varias entradas primarias, x_i y x_j, conmutan simultáneamente, las diferencias booleanas $\partial f/\partial x_i$, $\partial f/\partial x_j$ están indefinidas, puesto que para su cálculo se han considerado al resto de señales en reposo. Por tanto, para esta situación la ecuación 2.25 deja de ser válida. En consecuencia, es necesario considerar una extensión del modelo de manera que incluya tanto las correlaciones de las señales como las conmutaciones simultáneas de señales.

[22] Recuérdese que esto significa que una transición en la entrada x_i implica una transición en la salida f

Ghosh [46] propone el uso de simulación simbólica aplicando la XOR en dos tiempos consecutivos de la función de salida (*f*), en vez de aplicar la XOR respecto a los cambios en las entradas como hacía Najm en la anterior propuesta. Con esta propuesta se soluciona el efecto de las transiciones simultáneas. La ecuación propuesta es la siguiente:

$$f(t=0) \oplus f(t=T) = (f^0 \wedge \neg f^T) \vee (\neg f^0 \wedge f^T) \tag{2.26}$$

Que de forma simplificada se puede escribir como:

$$f^0 \oplus f^T \tag{2.27}$$

Debido a que se produce una conmutación cuando el valor de la señal es diferente en dos ciclos consecutivos, la probabilidad de la XOR de la función en dos ciclos consecutivos evalúa la actividad de conmutación:

$$a_f = P\{ f^0 \oplus f^T \} \tag{2.28}$$

En este trabajo se consideran también las dependencias estructurales debidas al *fan-out* reconvergente, los efectos de los retardos en las puertas (que producen transiciones espurias) y los elementos de memoria. Esto se verá en los apartados 2.3.4 y 2.3.5.

Para considerar las transiciones simultáneas que no se consideraban en la propuesta de Najm [90] (figura 2-24), **Chou** propone el uso de la diferencia booleana generalizada, implementándolo en la herramienta PAS [28]. Sin embargo, este método es muy intensivo en cómputo y la ecuación 2.28 propuesta por Ghosh es más sencilla y considera transiciones simultáneas.

Para manejar las dependencias temporales, **Schneider** [118], [119] modela el comportamiento de la señal como un proceso de Markov [50] estacionario en sentido estricto. Y de manera similar a Ghosh [46], calcula la actividad de transición evaluando la probabilidad de que la función cambie de valor en dos ciclos consecutivos $Tf = f^0 \oplus f^T$. Esta ecuación la llama *función de transición*. Utilizando esta función y mediante la expansión de *Shannon* (ecuación 2.13) se realizan los cálculos actividad.

La manera de realizar este cálculo se muestra en las siguientes ecuaciones[23], donde x^0 y x^T representan la entrada x para los tiempos 0 y T respectivamente, siendo T el ciclo del circuito. Mientras que $Tf_{x(0)}$ es el cofactor[24] de Tf respecto a x^0; $Tf_{x(0)x(T)}$ es el cofactor de $Tf_{x(0)}$ respecto a x^T.

$$\begin{aligned} a = P(Tf) &= P\{(x^0 \wedge Tf_{x(0)}) \vee (\neg x^0 \wedge Tf_{\bar{x}(0)})\} = \\ &= P\Big\{ (x^0 \wedge x^T \wedge Tf_{x(0)x(T)}) \vee (x^0 \wedge \neg x^T \wedge Tf_{x(0)\bar{x}(T)}) \vee \\ &\vee (\neg x^0 \wedge x^T \wedge Tf_{\bar{x}(0)x(T)}) \vee (\neg x^0 \wedge \neg x^T \wedge Tf_{\bar{x}(0)\,\bar{x}(T)}) \Big\} \end{aligned} \tag{2.29}$$

Por ser $\{x^0 \wedge x^T\}$; $\{x^0 \wedge \neg x^T\}$; $\{\neg x^0 \wedge x^T\}$; y $\{\neg x^0 \wedge \neg x^T\}$ eventos disjuntos y al asumir que las entradas son mutuamente independientes, la ecuación queda:

$$\begin{aligned} a = P(x_{1\to1})\cdot P\left(Tf_{x(0)x(T)}\right) + P(x_{1\to0})\cdot P\left(Tf_{x(0)\bar{x}(T)}\right) + \\ + P(x_{0\to1})\cdot P\left(Tf_{\bar{x}(0)x(T)}\right) + P(x_{0\to0}) \cdot P\left(Tf_{\bar{x}(0)\bar{x}(T)}\right) \end{aligned} \tag{2.30}$$

Esta fórmula separa x de la función de transición Tf. Cada cofactor de Tf no depende de x, y así se van separando el resto de variables aplicando el método recursivamente.

Las operaciones de la función de transición, Tf, se realizan mediante BDD. Ya que como se ha visto en la figura 2-12, la expansión de *Shannon* es traducible directamente a BDD. Consecuentemente, el orden de las variables es igual que el de la expansión de *Shannon*, donde a cada variable x^0 siempre le sigue x^T. Estos BDD que conservan esta dependencia temporal se han denominado TFBDD[25]. Este orden se debe a que x^0 y x^T son mutuamente dependientes, pero

[23] Recuérdese que se consideran procesos estrictamente estacionarios, por lo que el origen de tiempos es intranscendente

[24] Véase ecuación 2.14

[25] TFBDD: Diagrama de decisión binaria de la función de transición

independientes del resto de variables. Por esto, se evalúan conjuntamente, separándolas de las probabilidades resto.

Un ejemplo de la creación del TFBDD para la puerta OR se muestra en la figura 2-25. De la misma manera que los BDD, el TFBDD resultante se recorre por todos los caminos que van desde el nodo raíz, A^0, hasta el nodo final "1". Cada camino es disjunto del resto y dentro de cada camino solamente se evalúan conjuntamente los nodos de la misma variable en tiempos distintos. La ecuación de la actividad de conmutación resultante se muestra también en la figura. En la sección 3.5.1 se profundiza más en este tipo de BDD.

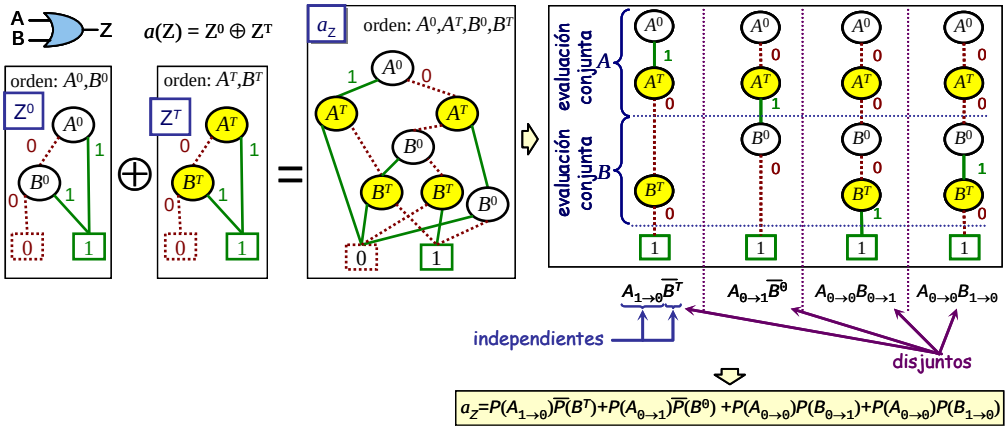


Figura 2-25: Creación del TFBDD para una puerta OR y obtención de la ecuación actividad

En los circuitos muy grandes, para evitar los inconvenientes de construir BDD globales que requerirían muchos recursos computacionales y de memoria, se propone dividir el circuito en zonas de reconvergencia estructuralmente independientes. Adicionalmente, de manera similar a la propuesta de las superpuertas (figura 2-21), se propone un cálculo aproximado para reducir el tamaño de las regiones de modo que las señales que superan un nivel de profundidad definido por un parámetro Δ quedan excluidas de la región de reconvergencia.

Bhanja [6] propone el uso de redes bayesianas (BN) para la elaboración del modelo de probabilidades del circuito. La ventaja de este tipo de representación gráfica es que no sólo hace explícita la dependencia condicional entre nodos, sino que sirve de eficiente mecanismo de cómputo para actualizar las probabilidades. Las dependencias estructurales y temporales son tenidas en cuenta. Para circuitos grandes tienen que dividir las redes bayesianas para evitar un excesivo uso de recursos computacionales y de memoria. El método propuesto de partición proporciona mejores resultados que la aproximación propuesta por Schneider [119]. En la tabla 4-9 se muestran los errores de estas propuestas.

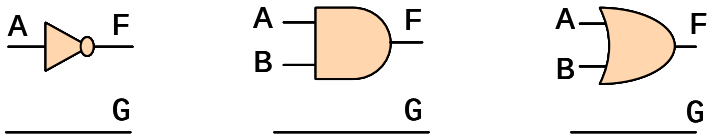
2.3.3 Modelos que consideran las correlaciones espaciales en las entradas

El primer modelo que tuvo en cuenta las correlaciones espaciales en las entradas fue propuesto por **Ercolani** [37]. Esta propuesta está orientada a los estudios de testabilidad y por tanto, el modelo sólo obtiene la probabilidad de señal y no la actividad. El modelo, para no tener que dividir el circuito en superpuertas, propaga las probabilidades de señal considerando coeficientes de correlación entre parejas de señales. Estos coeficientes de correlación se definen a partir de la ecuación:

$$P\{A \wedge B\} = P_A P_B SC_{AB} \tag{2.31}$$

Donde SC_{AB} es el coeficiente de correlación espacial entre las señales A y B. SC_{AB} es igual a 1 cuando A y B son independientes. Mediante fórmulas asociadas a cada puerta lógica, estos coeficientes se propagan de manera análoga a las probabilidades de señal. La figura 2-26

muestra las fórmulas de coeficientes de correlación espacial para distintas puertas lógicas, respecto a una señal cualquiera *G*.



$$SC_{FG} = \frac{1 - P_A SC_{FG}}{1 - P_A} \qquad SC_{FG} = SC_{AG} SC_{BG} \qquad SC_{FG} = \frac{P_A SC_{AG} + P_B SC_{BG} - P_A P_B SC_{AG} SC_{BG} SC_{AB}}{P_A + P_B - P_A P_B SC_{AB}}$$

Figura 2-26: Fórmulas de los coeficientes de correlación espacial asociados a puertas

Marculescu [80] considera dependencias espacio-temporales de primer orden en las entradas y dependencias estructurales debidas al *fan-out* reconvergente. De la misma manera que Schneider (§2.3.2, [119]), para manejar las correlaciones temporales de primer orden, la señal es modelada como un proceso de Markov estacionario en sentido estricto y de retardo unidad.

El avance más destacable de este artículo es la inclusión de las correlaciones espacio-temporales de primer orden de las entradas. La diferencia con el trabajo propuesto por Ercolani [37] es que en aquel trabajo las correlaciones eran sólo espaciales y no temporales. Definiendo el coeficiente de correlación de transición $TC_{x,y}^{ij,kl}$ se capturan estas dependencias entre pares de señales. Este coeficiente determina las correlaciones entre las señales x e y al efectuar las transiciones $i \rightarrow k$, $j \rightarrow l$, respectivamente durante dos ciclos de reloj consecutivos, donde $i, j, k, l \in \{0, 1\}$.

$$TC_{x,y}^{ij,kl} = \frac{P(x_{i\rightarrow k} \wedge y_{j\rightarrow l})}{P(x_{i\rightarrow k}) \cdot P(y_{j\rightarrow l})} \tag{2.32}$$

Y por tanto:

$$P(x_{i\rightarrow k} \wedge y_{j\rightarrow l}) = P(x_{i\rightarrow k}) \cdot P(y_{j\rightarrow l}) \cdot TC_{x,y}^{ij,kl} \tag{2.33}$$

Correlaciones de mayor orden son ignoradas, asumiendo que el coeficiente de correlación de transición entre más de dos señales es el producto de sus correspondientes correlaciones entre pares de ellas:

$$TC_{x,y,z}^{ijk,lmn} = \frac{P(x_{i\rightarrow l} \wedge y_{j\rightarrow m} \wedge z_{k\rightarrow n})}{P(x_{i\rightarrow l}) \cdot P(y_{j\rightarrow m}) \cdot P(z_{k\rightarrow n})} = TC_{x,y}^{ij,lm} \cdot TC_{x,z}^{ik,ln} \cdot TC_{y,z}^{jk,mn} \tag{2.34}$$

Para evitar los grandes BDD que surgen del uso de las entradas primarias proponen un enfoque incremental en el que no sólo las probabilidades de transición se propagan, sino que también los coeficientes de correlación de transición. Similarmente a la simplificación hecha por Schneider en las regiones de reconvergencia, definen un error relativo ε con el que limitan su profundidad, que en la práctica es menor que 8.

Bhanja [7] amplia su anterior trabajo [6] (§2.3.2) para considerar correlaciones dos a dos entre los puertos de entrada. Para considerar las correlaciones espaciales en las entradas utilizan el mismo mecanismo que el empleado en la partición de las redes bayesianas (BN) para simplificar el modelo.

2.3.4 Modelos que consideran los retardos en las puertas

El trabajo de **Ghosh** [46] (§2.3.2) también consideraba retardos en las puertas. Para ello emplean simulación simbólica, en donde utilizan tanto modelos de retardo unitario como de retardo variable.

Costa [31] aplica la simulación polinómica propuesta por Parker [103] a la actividad de conmutación. Consideran retardos en las puertas y con ello la aparición de transiciones espurias. Para ello, asignan a todas las salidas de las puertas lógicas una forma de onda de grupos de polinomios llamada *forma de onda polinómica* (*polynomial waveform*). Cada grupo representa las condiciones a la salida de la puerta en un instante particular.

Las dependencias estructurales son también tratadas recurriendo al concepto de superpuertas, limitando igualmente su tamaño a un nivel de profundidad. Para reducir el tamaño de los polinomios y la complejidad del método proponen sustituir las variables polinómicas por sus valores aritméticos. Debido a que con esto se pierde exactitud, proponen el concepto de *nodos activos*, que son aquellos que deben ser usados como variables en el polinomio de la salida de una puerta para así conservar las dependencias estructurales.

Theodoridis [129] aplica el concepto de *función booleana en tiempos* (TBF) [73] para considerar retardos reales de las puertas. El modelo también considera dependencias estructurales y temporales, y proponen la base teórica para considerar las correlaciones espaciales en las entradas. Elaboran un algoritmo para evaluar la actividad de conmutación en tiempos específicos empleando TBF-OBDD (diagramas de decisión binaria ordenados para TBF). El método es muy intensivo en cómputo, por ello debe simplificarse para circuitos grandes.

2.3.5 Modelos que consideran los elementos de memoria

La propuesta de **Ghosh** [46], que ya se ha citado en los apartados 2.3.2 y 2.3.4, también trata circuitos secuenciales. Para calcular las probabilidades de estado del circuito secuencial, se modela el sistema como un proceso de Markov de tiempo y transiciones discretas, y se resuelven las ecuaciones de Chapman-Kolmogorov (§2.2.3.5). Estas ecuaciones se obtienen a través del gráfico de transición de estados (STG). El inconveniente es que si hay N biestables, resultan 2^N ecuaciones a partir del gráfico de transición de estados. Por tanto, esto limita el tamaño de los circuitos que se pueden tratar. En consecuencia, proponen aproximaciones basadas en el conjunto de estados alcanzables y en los estados fuertemente conectados.

Tsui [133] profundiza en la estimación de circuitos secuenciales propuesta por Ghosh [46]. Ya que la resolución de las ecuaciones de Chapman-Kolmogorov (§2.2.3.5) resulta muy costosa para circuitos grandes, proponen varios métodos aproximados. Uno de ellos es el cálculo de las probabilidades tomando los bits independientemente, con ello, en vez de 2^N ecuaciones resultan N ecuaciones no lineales. El inconveniente es que el error es significativo. Para aumentar la precisión de esta aproximación proponen 2 métodos: unir los bits de los estados en grupos de m elementos y desenrollar la lógica secuencial.

La agrupación m bits de estados está a medio camino entre el cálculo exacto (un único grupo de N elementos, m=N) y el tomar los estados independientemente (N grupos de 1 elemento, m=1). El número de ecuaciones depende del tamaño de m, tomando un valor entre N y 2^N.

En el método de desenrollado de la lógica secuencial, el estado siguiente (NS) se desenrolla k veces (figura 2-27). Se puede construir un conjunto de ecuaciones no lineales correspondientes con esta red k-desenrollada. Estas ecuaciones capturan parcialmente las correlaciones entre los estados. El número de ecuaciones permanece constante (N).

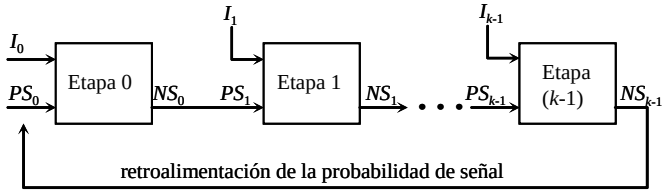


Figura 2-27: Desenrollado de la lógica secuencial

Para la resolución de este tipo de ecuaciones no lineales recurren a métodos numéricos como Newton-Raphson o Peano-Ricard.

Simultáneamente **Schneider** [120], [117] también propone desenrollar la lógica secuencial y además propone emplear sus propuestas en circuitos combinacionales y el uso de TFBDD [118] para considerar las dependencias temporales.

Para considerar las correlaciones temporales dentro del estudio de circuitos secuenciales, **Chou** desarrolla la herramienta PASS [26]. Ésta extiende el gráfico de transición de estados para considerar las correlaciones temporales, formando el ESTG (*Extended State Transition Graph*). En el ESTG, cada estado S_i del STG se separa en K estados, donde K es el número de vértices salientes del estado S_i. En consecuencia, si el circuito tiene M entradas primarias y N entradas de estados, el número de estados del ESTG será como máximo 2^{M+N}. Si ya para circuitos grandes la resolución de estas ecuaciones es complicada sin considerar las correlaciones temporales, con esta propuesta se aumenta exponencialmente la complejidad. En la figura 2-28 muestra el ESTG del ejemplo de la figura 2-18, en el que se puede apreciar el aumento de la complejidad.

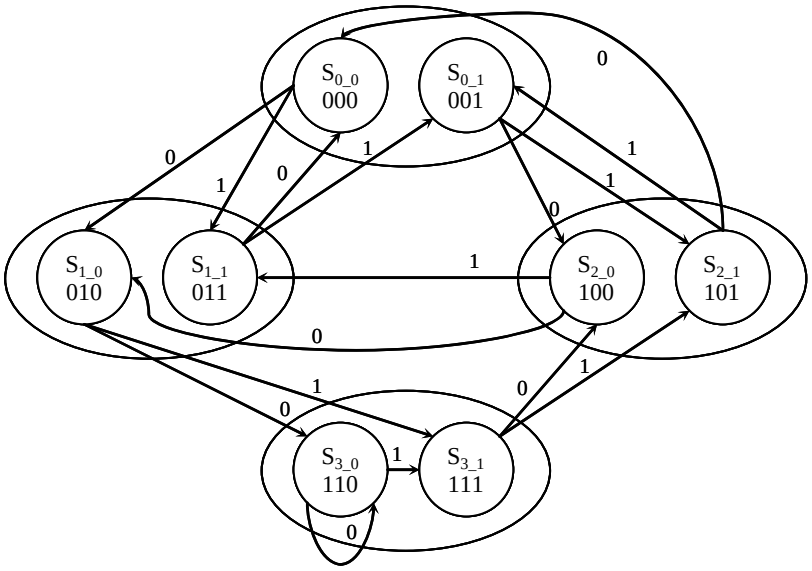


Figura 2-28: Gráfico de transición de estados extendido del ejemplo de la figura 2-18

Bhanja amplía sus trabajos previos ([6] en el apartado 2.3.2 y [7] en el apartado 2.3.3) para considerar elementos de memoria [5]. La propuesta crea una estructura de modelo de dependencia temporal (*temporal dependency model structure - TDM structure*) que explícitamente modela dependencias espaciotemporales entre las realimentaciones de los elementos de memoria. Esta estructura TDM es una red bayesiana dinámica (DBN). Una red bayesiana dinámica es una generalización de las redes bayesianas en la que consideran los efectos temporales de un conjunto de variables aleatorias cambiantes.

Debido a que las redes bayesianas son grafos dirigidos que no tienen bucles (DAG: *directed acyclic graph*), las líneas de realimentación no pueden volver a la misma variable de la red bayesiana. Para solucionar esto, se crean copias de la red bayesiana en distintos ciclos de reloj. Es similar a las técnicas que ya se han descrito de desenrollado de la lógica secuencial propuestas por Tsui [133] y Schneider [120]. Sin embargo, en esta propuesta, en vez de replicar sólo los elementos de memoria, replican el circuito entero. En la práctica proponen implementar tres instantes temporales.

Por último, para circuitos grandes, emplean un método de muestreo estocástico llamado muestreo lógico probabilístico (PLS) con el que infieren las probabilidades de las redes bayesianas de manera aproximada. Este método de inferencia no es simulativo. En los experimentos se utilizan 1000 muestras.

2.3.6 Modelos que consideran otros niveles de abstracción

La característica principal de los modelos expuestos en este apartado es que proponen realizar la estimación en niveles de abstracción superiores al nivel de puertas.

La propuesta de **Ferreira** [41] sigue el concepto de simulación polinómica (véase el apartado 2.3.7). Esta propuesta caracteriza los componentes de un circuito en nivel RT. La caracterización

se realiza sobre los componentes en nivel de puertas y a partir de ella se obtiene un polinomio de probabilidad. Los polinomios de probabilidad se van aplicando de módulo a módulo desde las entradas a las salidas del circuito RT. La propuesta considera los retardos de las puertas y emplean ZBDD (*zero suppressed BDD*) que ofrecen una representación compacta de la función lógica implementada.

Aunque esta propuesta estima la actividad en el nivel RT, realmente la estimación probabilística la realiza en nivel de puertas y emplea esa información para elaborar un modelo RT. Así que, desde el nivel de puertas elabora un macromodelo del componente por métodos probabilísticos, para luego realizar la estimación del circuito completo a partir de los macromodelos de los componentes.

Wright [139] afirma estimar la actividad de conmutación de circuitos combinacionales descritos en VHDL en nivel comportamental y puertas. El estimador está implementado en la herramienta BLAPE. La estimación es sin retardos y no consideran dependencias temporales (ecuación 2.21), ni correlaciones a las entradas. Proponen el uso de *BDD conectivos* (CBDD) [138] que tienen la ventaja de no aumentar tanto de tamaño como los BDD, aunque pierden la canonicidad y no conservan bien las dependencias estructurales.

El método transforma la descripción VHDL en un conjunto de ecuaciones booleanas, compuestas por funciones AND y OR. Las ecuaciones se transforman en una *netlist* a nivel de puertas en formato BLIF[26] con la que posteriormente se construye el CBDD. Los CBDD se pueden limitar en profundidad para reducir los recursos computacionales, disminuyendo con ello su exactitud.

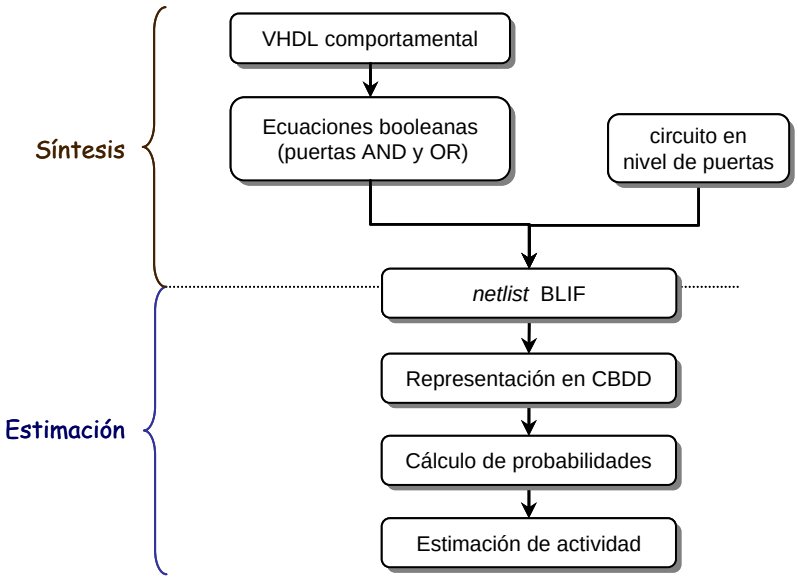


Figura 2-29: Flujo de estimación propuesto en BLAPE [139]

El flujo seguido en la estimación se muestra en la figura 2-29. En ella se puede apreciar que el método tiene dos partes diferenciadas:

- La primera recibe un circuito VHDL descrito en nivel comportamental y lo transforma en una *netlist* en formato BLIF, también puede recibir un circuito descrito en nivel de puertas. Es por tanto una síntesis.
- La segunda etapa realiza propiamente la estimación. Nótese que no existe ningún flujo de información desde el VHDL comportamental hacia el estimador, el estimador tiene como única entrada la *netlist* en nivel de puertas.

Por tanto, en nuestra opinión, según lo que se extrae de las referencias citadas, se trata de un estimador en nivel de puertas que previamente realiza una síntesis del VHDL comportamental.

[26] BLIF: *Berkeley Logic Interchange Format*

En el ámbito de la testabilidad, **Fernandes** [38] propone un método probabilístico para estimar la controlabilidad de circuitos RTL descritos en Verilog. Implementado en la herramienta ASCOPA, el método trata circuitos secuenciales resolviendo de manera aproximada las ecuaciones de Chapman-Kolmogorov que describen los estados del circuito (§2.2.3.5) y calculando la probabilidad asociada con cada estado de la cadena de Markov. Para reducir el esfuerzo computacional y tratar con diseños grandes empelan métodos de representación simbólica que describen la funcionalidad del circuito mediante BDD. Este trabajo constituye el primer método estático propuesto que de manera eficiente calcula las probabilidades de un circuito RTL.

2.3.7 Modelos a mitad de camino entre probabilísticos y simulativos

Este apartado muestra algunos modelos que están a mitad de camino entre los modelos probabilísticos y los simulativos.

Con el objetivo de analizar los fallos debidos electromigración en los buses de alimentación y tierra, **Najm** [93] introduce correlaciones temporales y realiza una simulación probabilística dirigida por eventos de las formas de onda esperadas. La propuesta se implementa en la herramienta CREST. Así, aunque con otro objetivo, la herramienta obtiene las probabilidades de transición del circuito. El algoritmo de propagación es muy similar a la simulación dirigida por eventos con modelo de retardos asignable, con la única diferencia que trabaja con probabilidades de señal y de transición. Así que es un modelo a mitad de camino entre el simulativo y el probabilístico, teniendo tiempos equiparables a las técnicas dinámicas.

En el modelo, las dependencias estructurales también están consideradas con el concepto de superpuertas. Las formas de ondas que se simulan indican la probabilidad de señal en distintos intervalos de tiempo y las probabilidades de transición de 0 a 1 en las fronteras de dichos intervalos. La figura 2-30 muestra un ejemplo de una forma de onda de probabilidad.

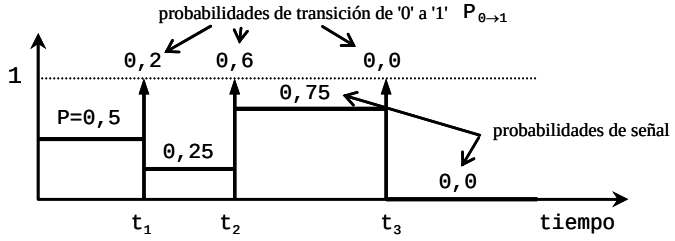


Figura 2-30: Forma de onda de probabilidad

Tsui [134] usa las formas de onda de probabilidad empleadas en CREST [93] y las extiende usando lo que llaman *simulación probabilística etiquetada*. Con ella tratan las correlaciones debidas a las reconvergencias. Para los cálculos emplean BDD. Los retardos de las puertas también son considerados.

Ya se citó en el apartado 2.3.4 que **Costa** [31] aplica la simulación polinómica (propuesta por Parker [103]) para calcular la actividad de conmutación. Con este modelo consideran retardos en las puertas y con ello la aparición de transiciones espurias.

Ampliando la simulación probabilística propuesta en CREST [93] y continuada por Tsui [134], **Hu** [57] usa las formas de onda de probabilidad considerando también los retardos de las puertas. La novedad de esta propuesta es que consideran la degradación que sufren las transiciones espurias (§1.3.3, [113]). Para los cálculos emplean TBF y con el objetivo de reducir su tamaño también dividen el circuito en superpuertas. El método resulta de media entre 6 y 8 veces más rápido que la simulación.

2.3.8 Resumen

Se han expuesto más de una veintena de técnicas probabilísticas y aunque hay más, se ha tratado de poner aquellas que se han considerado más significativas. En este resumen se han incluido también las técnicas basadas en simulación probabilística [93], [134], [57], que están a medio camino entre las técnicas probabilísticas y simulativas.

Las primeras técnicas probabilísticas que surgieron no estaban orientadas al cálculo de la actividad de conmutación, sino a la probabilidad de señal. Estos trabajos se han incluido debido a que el conocimiento de la probabilidad de señal es un paso previo para el cálculo de la actividad e incluso, en algunos casos, la actividad se puede deducir directamente a partir de la probabilidad. Independientemente del objetivo del trabajo, el fundamento de los métodos es similar y muchos de los trabajos posteriores se han basado o inspirado en ellos.

Los primeros métodos eran sencillos y paulatinamente se han ido complicando al incluir diferentes tipos de dependencias y circuitos. Desde el principio, el problema principal consistía en la elaboración de un modelo manejable que fuese suficientemente preciso. La consideración de las correlaciones estructurales, necesarias para lograr unos resultados fiables, hacía inviable la elaboración del modelo probabilístico para determinados circuitos y por ello se desarrollaron **técnicas de particionado**. La mayoría de las veces, estas particiones implican una **aproximación** del resultado (señalados con "~✓" en la tabla 2-2). Estas particiones restan fiabilidad al modelo a cambio de un menor uso de recursos computacionales y de memoria. Sólo se ha visto que Agrawal [2] haya llevado a cabo un tipo de análisis que divida el circuito sin recurrir a aproximaciones. Debido a que su análisis estaba limitado al cálculo de la probabilidad, para la actividad, su partición también sería inexacta. Por otro lado, el método partición propuesto implica un cierto coste computacional.

Los modelos se han perfeccionado, incluyendo las dependencias temporales, retardos en las puertas, dependencias en las entradas y elementos de memoria. **No se ha encontrado ningún método probabilístico que incluya todas estas características**. En el nivel de puertas lógicas el más completo es el propuesto por Bhanja [5], al que sólo le faltaría incluir los retardos en las puertas.

Casi todos estos métodos trabajan en el nivel de puertas lógicas. Hay muy **pocos** que hayan subido al **nivel RT**. La aportación de **Wright** [139] considera circuitos descritos en VHDL comportamental, pero en realidad, en nuestra opinión, lo que hace es sintetizar primero el VHDL al nivel de puertas y posteriormente realizar la estimación en el nivel de puertas.

Ferreira [41], estima la actividad en el nivel RT, pero realmente la estimación probabilística la realiza en nivel de puertas y emplea esa información para elaborar un modelo RT. Así que, desde el nivel de puertas elabora un macromodelo del componente por métodos probabilísticos, para luego realizar la estimación del circuito completo a partir de los macromodelos de los componentes.

Fernandes [38] es el único que estima mediante métodos probabilísticos la probabilidad en el nivel RT, centrándose además en circuitos secuenciales. Debido a que es un método orientado a la testabilidad, la actividad de conmutación no está considerada.

A fin de facilitar la comparación entre las distintas técnicas propuestas, se ha incluido la tabla 2-2 que las resume. En esta tabla las propuestas están ordenadas por fecha de aparición. La casilla marcada con un "✓" indica que el método considera el aspecto que indica su columna. Mientras que cuando está marcada con un "~✓" indica que lo considera de manera aproximada. La casilla vacía señala que ese aspecto no está considerado.

	Dep. Estructurales	Actividad	Dep. temporales	Transiciones simultáneas	Retardos	Dep. espaciales a las entradas	Elementos de memoria	RTL
Parker'74 [103]	~✓							
Seth"85 [121]	~✓							
Cirit'87 [29]		✓						
Najm'88 [93]	~✓	✓						
Erolani'89 [37]	~✓					~✓		
Najm'91 [90]	~✓	✓	✓					
Ghosh'92 [46]	~✓	✓	✓	✓	✓		✓	
Tsui '93 [134]	~✓	✓	✓	✓	✓			
Chou'94 [28]	~✓	✓	✓	~✓				
Schneider'94 [118]	~✓	✓	✓	✓				
Marculescu'94 [80]	~✓	✓	✓	✓		~✓		
Tsui'95 [133]	~✓	✓		✓			✓	
Schneider'95 [120]	~✓	✓	✓	✓			✓	
Chou'95 [26]	~✓	✓	✓	✓			✓	
Costa'97 [31]	~✓	✓	✓	✓	✓			
Agrawal"98 [2]	✓							
Ferreira'00 [41]	~✓	✓	✓	✓	✓			~✓
Theodoridis'00 [129]	~✓	✓	✓	✓	✓			
Wright'01 [139]	~✓	✓						~✓
Bhanja'01 [6]	~✓	✓	✓	✓				
Bhanja'02 [7]	~✓	✓	✓	✓		✓		
Fernandes'04 [38]							✓	✓
Bhanja'05 [5]	~✓	✓	✓	✓		✓	✓	
Hu'05 [57]	~✓	✓	✓	✓	✓			
Esta tesis	✓ / ~✓	✓	✓	✓				✓

Tabla 2-2: Propuestas en la estimación probabilística y aspectos considerados

De la tabla se puede observar que no hay ninguna propuesta que estime la actividad considerando de manera exacta las dependencias estructurales. Ni tampoco hay ninguna propuesta que calcule la actividad desde el nivel RTL.

2.4 Situación de la tesis en este contexto

En este capítulo se han expuesto los dos métodos para estimar la actividad de conmutación: el método dinámico y el método estático. La mayoría de los métodos estáticos recurren a técnicas probabilísticas.

En esta tesis se ha optado por las **técnicas probabilísticas,** las razones de esta elección son:

- Los menores tiempos de estimación que proporcionan los métodos estáticos.

- Continuar con la línea de herramientas CAD de análisis estático desarrolladas en el departamento.
- Los métodos estáticos se puede emplear sin haber finalizado el diseño y sin necesidad de tener el banco de pruebas.
- No se han propuesto métodos probabilísticos para la estimación de la actividad en RTL.

El trabajo de esta tesis se encuadra dentro de un conjunto de métodos y herramientas que ayudan al diseñador a mejorar la calidad y reducir el tiempo de diseño de los circuitos digitales. Este conjunto de métodos y herramientas se aglutinan en Ardid [132], una herramienta de diseño asistido por ordenador (CAD) que facilita la tarea de diseño orientado fundamentalmente en el nivel de transferencia de registros. Ardid ofrece un entorno de desarrollo integrado y asiste al diseñador ofreciéndole indicaciones sobre descripciones que comprometan la calidad del diseño. Gracias al modelo de los circuitos que crea Ardid (SHM [131]), se llevan a cabo análisis de calidad [130] y área [76]. El objetivo es continuar con esta línea de investigación y el análisis probabilístico llevaría a la elaboración de un modelo más completo que ayudaría no sólo al estudio de la actividad sino a crear y ampliar otros análisis, como los análisis de calidad [20], [132] y testabilidad [21].

Una de las principales diferencias entre las técnicas dinámicas y estáticas es que las técnicas dinámicas no requieren un análisis profundo del circuito. Por tanto, estas técnicas son aplicables a un amplio rango de niveles de descripción.

Sin embargo, las **técnicas estáticas o probabilísticas**, que son el objetivo de esta tesis, sí que **están condicionadas por el nivel de descripción** y no pueden adaptarse de un nivel de descripción a otro tan fácilmente como las dinámicas. Las técnicas probabilísticas que se han analizado están limitadas al nivel de puertas lógicas (§2.3). Como se ha dicho, la única técnica probabilística que estima en RTL [38] está orientada a la testabilidad y por tanto no estima la actividad de conmutación. Por otro lado, aunque la propuesta de Wright [139] considera el nivel RT, estrictamente, la estimación probabilística se realiza en el nivel de puertas. Por tanto, **no se han encontrados técnicas probabilísticas que estimen la actividad de conmutación en RTL**.

Esta tesis pretende dar un paso más hacia la estimación probabilística de la actividad de conmutación, proponiendo un modelo que estime en el nivel RT. Y sobre todo **aprovechar y demostrar las ventajas que este nivel puede aportar a la simplificación del modelo**. Subir del nivel de puertas lógicas al nivel RT tiene tres implicaciones esenciales:

- Cuanto más alto sea el nivel de abstracción de la estimación, ésta puede ser llevada a cabo en etapas anteriores del flujo de diseño. Esto proporciona información para la **toma temprana de decisiones que eviten iteraciones en el flujo de diseño** (§1.2 y §1.3.2).
- Las estimaciones a mayor nivel son **más sencillas y rápidas**, sin embargo a costa de una **pérdida de exactitud** por la falta de información sobre la implementación final.
- Realizar una estimación en un nivel más alto permite analizar el circuito de una manera más global que favorece una **estimación más eficiente**, pues las señales y los bloques se agrupan por funcionalidad y existe una información de más alto nivel que se pierde con la síntesis. Esto podría ser aprovechado para simplificar el método de estimación.

Este último punto es fundamental en esta tesis doctoral y sus beneficios se demostrarán a lo largo del documento.

Dentro de la estimación probabilística orientada a circuitos digitales diseñados en el nivel RT y puertas (y en concreto VHDL), los objetivos de esta tesis son:

- Elaborar el modelo probabilístico para la estimación de actividad de conmutación de las señales de circuitos digitales descritos en nivel RT.
- Aprovechar las características que el nivel RT ofrece para simplificar el modelo y así realizar una estimación más ventajosa que en el nivel de puertas.

- Proponer mecanismos más eficientes para el cálculo de la actividad de conmutación de las señales del circuito, ya que la complejidad de su cálculo incrementa extraordinariamente respecto a la de la probabilidad de señal.

Objetivos secundarios de esta tesis son:

- Desarrollar una herramienta automática que a partir de circuitos digitales descritos en VHDL-RTL elabore el modelo probabilístico y estime la actividad. Esta herramienta servirá como demostradora del modelo propuesto.
- Contribuir al análisis de la calidad de los diseños a través del modelo probabilístico.
- Integración de estas herramientas de estimación y calidad en Ardid [132], ampliando así su funcionalidad.
- Contribuir a otros análisis de los diseños, como lo serían, estimación de consumo, área, testabilidad, ...

Debido a que esta tesis es una primera aproximación a la estimación de actividad por medios probabilísticos en RTL, no se pretende elaborar un método que abarque todas las características enunciadas en la tabla 2-2. Como se verá en el capítulo siguiente, los aspectos más importantes que no se han considerado son los retardos en las puertas, las correlaciones espaciales y los circuitos secuenciales.

En el capítulo siguiente se presenta el método propuesto y posteriormente, en el capítulo 4, se muestran los resultados experimentales de la propuesta de estimación.

3. PROPUESTA DE ESTIMACIÓN DE LA ACTIVIDAD DE CONMUTACIÓN EN RTL

En este capítulo se expone la propuesta para la estimación probabilística de la actividad de conmutación en el nivel RT. Este trabajo es una primera aproximación hacia la estimación probabilística en este nivel y por lo tanto no aspira a cubrir todos sus aspectos, pero tal como esta tesis pretende demostrar, estas aportaciones son suficientes para considerar favorable la estimación RT frente la estimación en el nivel de puertas.

En el primer capítulo se expusieron las **ventajas metodológicas** la estimación en niveles de abstracción superiores frente a la estimación en niveles de abstracción más bajos. Sin embargo, también hay otras **ventajas prácticas** que afectan a la implementación del método de estimación. Todas estas ventajas se pueden resumir en:

- **Ventajas relativas a la metodología del diseño electrónico**: Cuanto más alto sea el nivel de abstracción en el que se pueden realizar las estimaciones, antes, en el flujo de diseño, se pueden prever las características del futuro circuito. Consecuentemente se logra una anticipación que permite predecir el cumplimiento de las especificaciones y detectar errores, lo que ayuda a tomar decisiones que evitarán los costosos bucles en el flujo del diseño (§1.2 y 1.3.2).
- **Ventajas relativas a la sencillez del diseño**: Las estimaciones en los niveles de abstracción superiores son más rápidas y sencillas que en los inferiores. Un diseño de más alto nivel tiene menos información de detalle que el mismo diseño en un nivel inferior, esto permite simplificar las estimaciones ya que gran parte de ese detalle no es necesario para la estimación. Sin embargo, dicha sencillez también provoca que las estimaciones sean menos exactas. Esto también se debe a que hay decisiones de diseño que aún no han sido tomadas, pero en muchos casos, son precisamente los resultados de la estimación los que contribuirán a esta toma de decisiones.
- **Ventajas relativas a la información de alto nivel**: Aunque un diseño descrito en un nivel más alto tenga menos detalle que el diseño en un nivel inferior, el diseño descrito en un nivel superior tiene información que se perderá en las tareas de síntesis. Esta información puede ser de gran utilidad a la hora de realizar las estimaciones y de simplificar los modelos. Cuanto más alto sea el nivel de descripción, más fácil resulta comprender la funcionalidad del diseño, y con ello, extraer información útil para la estimación.

A partir de las citadas ventajas sería razonable concluir que la estimación de la actividad de conmutación en el nivel RT es ventajosa frente a la estimación en el nivel de puertas. Sin embargo, como se ha visto en el capítulo anterior (§2.3), a pesar de que la estimación probabilística de la actividad de conmutación ha sido muy prolífica en cuanto a aportaciones, ha estado restringida al nivel de descripción de puertas. De la gran cantidad de referencias consultadas, no se ha encontrado ninguna que estime mediante métodos probabilísticos la actividad de conmutación en RTL. Ya se ha comentado que las referencias [41] y [139] afirman realizar la estimación en RTL y comportamental respectivamente, y aunque estiman el consumo de circuitos en este nivel, en ambos casos, el método probabilístico de estimación es aplicado en el nivel de puertas. Por otro lado Fernandes [38] es el único que se ha encontrado que realice estimación en RTL de la probabilidad de señal, pero no de la actividad, ya que su trabajo está orientado a la testabilidad.

En esta tesis **se propone realizar la estimación de la actividad de conmutación en el nivel RT**, para ello **se propone un método de estimación,** y con éste, se pretenden **demostrar las ventajas prácticas de la estimación en RTL frente al nivel de puertas**. Es decir, se mostrarán las diferencias en cuanto a rapidez y sencillez con suficiente precisión entre la estimación RTL comparadas con la estimación en puertas. También se pretenden demostrar las ventajas en cuanto a la información disponible en el nivel RT frente al nivel de puertas. Las otras ventajas de la estimación en RTL frente a puertas son las relativas a las metodologías de diseño de circuitos. Estas ventajas quedan demostradas por sí mismas, ya que, por las razones expuestas

en los apartados 1.2 y 1.3.2, la estimación es más ventajosa cuanto antes en el flujo del diseño se pueda estimar, y por tanto, cuanto más alto sea el nivel abstracción de los circuitos que se pueden estimar.

La propuesta de estimación es amplia, e incluye tanto aspectos metodológicos y generales como aspectos particulares que resuelven cuestiones concretas. Por ello, y a fin de facilitar la comprensión de la propuesta, en el primer apartado de este capítulo (§3.1) se ha elaborado un resumen que a modo de guión esboza las diferentes propuestas de esta tesis. El resto de apartados harán referencia a este guión para situarlo dentro del contexto general. Algunos apartados se refieren a propuestas particulares y pueden tener un gran nivel de detalle. En el caso en el que el lector, se sienta perdido, se recomienda que acuda al primer apartado y lo sitúe en el contexto.

No obstante, al ser una primera propuesta, este trabajo no puede abarcar todos los aspectos del nivel RT, y por tanto el modelo probabilístico de actividad es limitado. En la sección 3.2 se detallarán las características y las simplificaciones realizadas para este modelo.

Una vez expuestas las consideraciones para la realización del modelo, se dedicará la sección 3.3 a la descripción del análisis estructural del circuito y la elaboración del modelo hardware del circuito. Este modelo se construye como base del modelo probabilístico y es necesario en el nivel RT, ya que se precisa de una estructura o grafo de dependencias e influencias de los nodos internos. Esto lo diferencia de las otras propuestas, ya que al realizar la estimación en el nivel de puertas disponen de la estructura exacta del circuito.

Posteriormente, la sección 3.4 explicará el análisis de reconvergencias y el método de partición propuesto. La sección 3.5 expondrá la construcción de los BDD y las ventajas de construirlos desde el nivel RT. En esta sección habrá una especial dedicación a los BDD de actividad, pues su tamaño aumenta de manera considerable respecto a los de probabilidad. En ella se propone una nueva forma de representar los BDD de actividad. La sección 3.6 muestra cómo el análisis del circuito desde el nivel RT permite producir un orden favorable de las variables de los BDD. La sección 3.7 explica el método de propagación de las probabilidades y actividades a través del circuito. Y por último, en la sección 3.8 se exponen las conclusiones del capítulo.

3.1 Visión general de la propuesta de estimación de la actividad de conmutación

Ya se ha visto que existen dos tipos de métodos muy diferenciados para estimar la actividad de conmutación de los puertos de salida y nodos internos de un diseño digital: por un lado están los métodos simulativos, dinámicos o estadísticos, y por otro lado, los métodos estáticos o probabilísticos.

Los **métodos simulativos** realizan simulaciones del diseño con un conjunto largo de vectores de test y observan su comportamiento extrayendo las estadísticas de actividad de sus nodos. Estos métodos requieren largas simulaciones para reducir la dependencia de los resultados con los vectores de test aplicados. A esto se le añade que un cambio en las condiciones de las entradas hace necesario repetir todo el proceso de estimación. Este tipo de método es inviable para circuitos grandes y debido a ello se han realizado diversas propuestas que reducen el tamaño de los vectores y con ello el tiempo de simulación (§2.1.1). Por otra parte, la aplicación de este método es casi inmediata y trasladable a casi cualquier tipo de descripción, ya que sólo se requiere un simulador que permita extraer las estadísticas del comportamiento del circuito.

Los **métodos estáticos** elaboran modelos matemáticos del diseño con los que propagan los valores de probabilidad y actividad de las entradas a las salidas. Los modelos matemáticos más utilizados están basados en la **Teoría de la Probabilidad**, aunque existen otros basados en la Teoría de la Información y otros en la complejidad. Estos métodos son computacionalmente intensivos ya que la elaboración del modelo matemático resulta compleja si no se quiere perder precisión en la estimación. Sin embargo, al contrario de los métodos simulativos, una vez

elaborado el modelo del diseño, la estimación de la actividad para otras condiciones es sencilla y relativamente rápida. Este tipo de métodos es muy dependiente del nivel de abstracción del diseño, y por tanto, los modelos matemáticos están restringidos a un tipo de nivel de abstracción, ya que el cambio de nivel de abstracción conlleva un cambio en la consideración de los elementos del modelo y sus relaciones, tales como señales, operadores, transiciones y tiempos, entre otros.

En esta tesis se ha optado por el método estático probabilístico. Las razones de escoger el método estático son:

- Los menores tiempos de estimación.
- Las amplias posibilidades en cuanto al análisis de los diseños que ofrece la elaboración de un modelo del circuito. Esto permite continuar con las líneas de investigación del departamento.
- Los métodos estáticos se puede emplear sin haber finalizado el diseño, y sin necesidad de tener el banco de pruebas.
- La falta de modelos probabilísticos de actividad en el nivel RT.

Referente al segundo punto, la elaboración de un modelo del circuito permite un análisis que no sólo se limita al ámbito de la actividad de conmutación, sino que otros análisis pueden llevarse a cabo, como los análisis de testabilidad, consumo, calidad y área. Esto se enmarca en las líneas de investigación del departamento referentes al desarrollo de herramientas de ayuda al diseño.

Respecto al cuarto punto, ya se ha visto que existen multitud de propuestas para la estimación de la actividad de conmutación por medios probabilísticos, sin embargo, **todas ellas se restringen al nivel de puertas**[27]. La propuesta de Fernandes [38] realiza estimación probabilística en el nivel RT, aunque los resultados se limitan a la estimación de la probabilidad de las señales, sin extenderlo a la actividad de conmutación. Consecuentemente **no se han encontrado aportaciones en la estimación probabilística de la actividad de conmutación en el nivel RT**.

Por tanto, **en esta tesis se propone de manera original la elaboración de un modelo probabilístico para la estimación de la actividad de conmutación en el nivel RT**. Esto supone una novedad respecto con las anteriores propuestas, ya que, como se ha comentado, hasta ahora estos modelos se restringían al nivel de puertas.

La estimación en el nivel RT tiene las siguientes ventajas sobre la estimación en el nivel de puertas:

- Poder realizar la estimación en el nivel RT en vez de en puertas implica que se puede estimar antes en el flujo de diseño. Esto hace que el diseñador disponga con anterioridad de una información que le permite prever las implicaciones de su diseño y tomar decisiones que le **evitarán bucles en el flujo de diseño**.
- La elaboración del modelo probabilístico desde el nivel RT se simplifica respecto al nivel de puertas debido al menor grado de detalle del RTL.
- Además, el nivel RT proporciona una perspectiva y una información del diseño que resulta difícil de obtener en el nivel de puertas. Como se verá en el desarrollo de este capítulo, esta perspectiva permite la **elaboración de un modelo más sencillo**.

Sin embargo, la estimación en el nivel RT también tiene desventajas respecto al nivel de puertas:

- En el diseño en el nivel de puertas se dispone de la estructura más cercana a la del circuito final: cada nodo en el diseño de puertas existirá en el diseño final y las relaciones entre los nodos serán las mismas que las relaciones del circuito final. Sin embargo, en el nivel RT no

[27] Ya se ha comentado el caso de la referencia [139], en la que se asegura que estiman en el nivel RT. Sin embargo, en nuestra opinión, a partir de lo que se extrae de la citada referencia, los autores realizan una síntesis en el nivel de puertas y realizan la estimación a partir del diseño en puertas.

es seguro que todos los nodos vayan a existir en el diseño final, y además, en el circuito final aparecerán nuevos nodos que no existen en el nivel RT[28].

- Por tanto, en el nivel RT es necesario **crear una estructura genérica del circuito** (en el nivel de puertas era idéntica al circuito). Esta estructura será una aproximación del circuito en puertas, ya que el circuito final dependerá de la herramienta de síntesis utilizada, de las opciones de síntesis y la biblioteca escogida.
- Debido a que en el nivel RT no se conoce la disposición final de las puertas del circuito ni el tipo de puertas utilizadas, **resulta muy difícil realizar una estimación de las transiciones espurias** (*glitches*) debidas a la lógica combinacional.

Una vez definida la propuesta de esta tesis, con sus ventajas y limitaciones respecto a los métodos existentes, se va a proceder a esbozar el método de estimación propuesto. Cada una de las partes de este método será explicado con detalle en sus correspondientes apartados.

Ya se ha comentado que la elaboración de un modelo probabilístico del diseño en el nivel RT requiere la creación de una estructura genérica del circuito debido a que no se dispone del diseño en puertas final. Por tanto, en esta tesis **se propone la creación de una estructura genérica sobre la que se sustenta el modelo probabilístico**. Esta estructura genérica se ha denominado EHM (*Extended Hardware Model*: **modelo del hardware** extendido) y posibilita la elaboración de las relaciones entre los nodos de un circuito. Todas las sentencias y expresiones del RTL original son analizadas y traducidas al EHM.

Cuando se ha elaborado el EHM del circuito, se tiene de un modelo similar (más sencillo pero menos exacto) al que se tendría en el nivel de puertas. Sin embargo, con el EHM se dispone además de toda la información del nivel RT, lo que, como se demostrará, **permite llevar a cabo análisis y simplificaciones que son muy difíciles de realizar exclusivamente desde el nivel de puertas**. Por tanto, en esta tesis se propone realizar el análisis relativo al nivel de puertas con la información del nivel RT para así simplificar el modelo probabilístico.

Los modelos probabilísticos de circuitos grandes tienden a crecer excesivamente en complejidad y tamaño. Para evitar esto se recurre a realizar particiones del circuito que simplifican el modelo. Sin embargo, como ya se ha visto (§2.2.3.1), las reconvergencias de las señales impiden que el circuito se pueda dividir sin perder exactitud en la estimación tanto en los valores de probabilidad como de actividad de los nodos. Así, si se quiere mantener la exactitud, el circuito se debe partir en las llamadas regiones reconvergentes, que son aquellas que contienen todos los caminos de las señales reconvergentes (§2.3, figura 2-20, pág. 37).

La división del circuito en zonas reconvergentes es una práctica habitual en la elaboración de los modelos probabilísticos en el nivel de puertas. **En el nivel RT** la división se realiza de manera similar, aunque **es sensiblemente más sencilla** de hacer debido a que el circuito tiene menor información de detalle. Además, en algunos casos **la síntesis del diseño en el nivel de puertas puede hacer aumentar la reconvergencia impidiendo la partición del circuito**, mientras que el mismo circuito en RTL es divisible en regiones reconvergentes (por ejemplo el circuito de §4.2.2). Es por tanto **una ventaja de la estimación RTL frente al nivel de puertas**. El proceso de partición del EHM en zonas reconvergentes está explicado en el anexo, apartado AII.1.

Sin embargo, la partición del circuito en zonas reconvergentes no siempre soluciona el problema de la complejidad y tamaño del modelo probabilístico. En algunos circuitos las reconvergencias de las señales son tan grandes que no permiten partición alguna, o el tamaño de las zonas reconvergentes es tal que las particiones siguen produciendo modelos grandes y complejos. Este problema se presenta tanto para el nivel de puertas como para el nivel RT y obliga a realizar otras simplificaciones que merman la exactitud del resultado.

Para evitar el excesivo tamaño del modelo probabilístico de las regiones reconvergentes, en esta tesis **se propone realizar además particiones por regiones disjuntas** (§3.4) para calcular la actividad y probabilidad. En muchas ocasiones una región reconvergente se puede dividir en

[28] De la misma manera que el circuito en puertas es menos exacto que el circuito descrito en transistores

regiones disjuntas más pequeñas. Las regiones disjuntas permiten calcular la **probabilidad** de manera **exacta** y la **actividad** de manera **aproximada**, mientras que otros métodos propuestos calculan de manera aproximada tanto la actividad como la probabilidad. Se debe señalar que es importante tener valores de probabilidad exactos ya que la actividad de un nodo se calcula en función de la actividad y de la probabilidad de los nodos de los que depende.

La partición por regiones disjuntas se ha utilizado para la estimación de probabilidad en el nivel de puertas[29], sin embargo, la identificación de las regiones disjuntas es muy costosa en este nivel, teniéndose que recurrir a métodos simulativos para su identificación. Además, el método no asegura la identificación de todas las zonas. En esta tesis **se propone la identificación de las zonas disjuntas desde el nivel RT**, ya que la **identificación** es **inmediata** para algunas estructuras típicas de este nivel, como son los multiplexores y las sentencias *if* y *case*. Esta identificación se realiza desde el EHM con la información que tiene del nivel RT, y se explicará en el apartado 3.4.1.

La división por regiones disjuntas para el cálculo de la actividad es una aportación original de esta tesis. Debido a que esta división no produce resultados exactos para la actividad, en esta tesis **se han analizado las condiciones que minimizan el error cometido**. Esto ayuda a realizar particiones que produzcan errores pequeños, permite estimar la magnitud de este error y por tanto, evaluar en cada caso si es aceptable la división. Este análisis se desarrolla en el apartado 3.4.3 y se verifica en los resultados experimentales.

Con el EHM dividido en regiones reconvergentes y en regiones disjuntas se pueden elaborar las funciones que calculan los valores de la probabilidad y actividad de los nodos del circuito. Para cada nodo del circuito se construye su ecuación lógica en función de otros nodos pertenecientes a su misma partición. La evaluación de esta función lógica proporciona los valores de probabilidad y actividad del nodo. Existen diversas representaciones para la construcción y el manejo de las funciones, como puede ser en forma polinómica, diagramas de decisión binaria (BDD) o redes bayesianas. En esta tesis se han escogido los BDD, por su amplio uso y disponibilidad de herramientas de libre distribución y código abierto, lo que ha sido decisivo para poder implementar las modificaciones propuestas.

La evaluación de la actividad se puede realizar de diversas maneras, la más extendida, y que proporciona mayor exactitud, es aplicar la XOR de la función de probabilidad del nodo en dos tiempos consecutivos (véase la ecuación 2.28, página 40). La función resultante es mucho más complicada y de mayor tamaño que la de probabilidad inicial, lo que dificulta aún más el modelo, y hace necesario recurrir a simplificaciones y optimizaciones.

Debido a que la representación de la actividad es bastante más complicada que la de la probabilidad, y con ello también el BDD de actividad, en esta tesis **se propone de manera original una nueva forma de representar los BDD de actividad**. Esta nueva representación (*a*BDD) aprovecha algunas características de la ecuación de actividad para representar los BDD de actividad de manera que reduzcan el número de nodos entre un 25 y un 50%. Las características de estos BDD propuestos están desarrolladas en el apartado 3.5.2, en donde también **se definen las operaciones que obtienen un *a*BDD a partir de su BDD** de probabilidad. También **se establecen las operaciones que transforman el BDD de actividad utilizado en otras propuestas en el *a*BDD propuesto**. Todas estas definiciones, operaciones y transformaciones constituyen aportaciones originales de la tesis.

Una última ventaja analizada en esta tesis referente al estudio de la actividad en el nivel RT tiene que ver con los tamaños de los BDD. El tamaño de los BDD (tanto de probabilidad como de actividad) es fuertemente dependiente del orden de las variables escogido, habiendo enormes variaciones de tamaño para una misma función. Aunque existen multitud de algoritmos propuestos para tratar de encontrar tamaños mínimos, la aplicación de estos algoritmos es costosa, aún así, en muchas ocasiones su aplicación merece la pena por las reducciones de tamaño logradas. Al realizar el análisis en el nivel de puertas el orden escogido es aleatorio y por tanto puede obtenerse un orden muy desfavorable. Por el contrario, **desde el**

[29] Véase el trabajo de Agrawal, apartado 2.3 página 37

nivel RT se pueden escoger órdenes suficientemente optimizados que permiten con poco esfuerzo reducir el tamaño de los BDD frente a los del nivel de puertas. En esta tesis **se propone la elección de un orden de variables del BDD basado en la funcionalidad de las señales en el nivel RT**. Esta elección del orden es *natural* en RTL y por tanto no implica un consumo adicional de recursos en la obtención de un orden apropiado. Este método se ha desarrollado empíricamente y está explicado en el apartado 3.6.

Además, para operaciones del nivel RT con vectores de más de un bit, como pueden ser la suma, diferencia, comparación, ... , se puede preestablecer un orden óptimo de las variables según el índice del vector. La detección de estas operaciones y de los índices de los vectores puede ser inmediata desde el nivel RT, sin embargo, no es fácil de realizar en el nivel de puertas. **El preestablecimiento de los órdenes de las variables para los operadores RTL supone una ventaja más para la estimación en el nivel RTL**. En el capítulo de los resultados experimentales se muestran ejemplos que corroboran estas ventajas.

Finalmente, una vez elaborado el EHM con los BDD de probabilidad y actividad del diseño se dispone ya del modelo probabilístico del diseño que permite calcular las probabilidades y actividades de las señales del circuito. A pesar de las simplificaciones propuestas, este modelo puede ser costoso de realizar para circuitos grandes. Sin embargo, se trata de un coste inicial, ya que una vez elaborado el modelo, puede ser utilizado para diferentes condiciones de entrada, teniendo únicamente que propagar los valores numéricos. Esto es bastante más rápido que la elaboración del modelo, y constituye una diferencia fundamental con los métodos simulativos, que requieren la realización de todo el proceso de simulación completo cuando se varían las condiciones de las entradas.

Para facilitar la comprensión del método de estimación propuesto, en la figura 3-1 se muestra un esquema general. En la figura se ha marcado con una estrella las aportaciones originales de esta tesis.

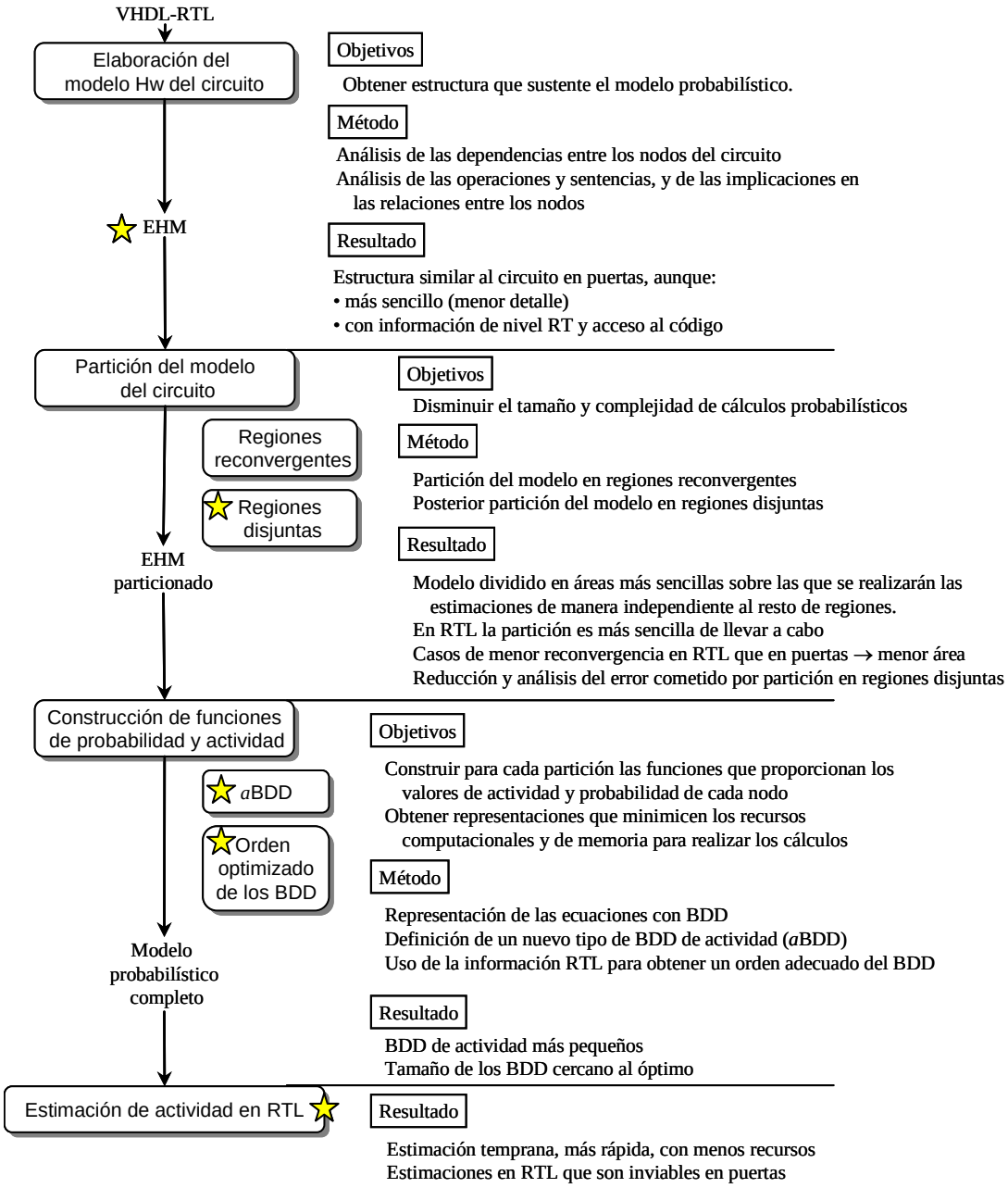


Figura 3-1: Esquema de la propuesta de estimación de esta tesis

3.2 Características del modelo propuesto

La estimación probabilística de la actividad de conmutación en RTL que se propone en esta tesis constituye el primer trabajo que afronta la estimación en RTL, y por tanto se han realizado diversas simplificaciones para facilitar el proceso. Estas simplificaciones no le restan generalidad, más bien al contrario, pues la inclusión del resto de aspectos corroboraría, en su mayor parte, las ventajas de la estimación RTL.

Las simplificaciones del modelo propuesto son:

- Sólo se consideran circuitos combinacionales
- El modelo no tiene retardos, por tanto no se consideran las transiciones espurias

- Se considera que hay un periodo o ciclo mínimo del circuito dentro del cual las señales no sufrirán transiciones. Este periodo es similar al que tendrían las señales registradas en un circuito secuencial.
- Por tanto, la suposición de este ciclo mínimo y la no consideración de los retardos hace que la actividad de conmutación pueda ser considerada como una probabilidad, ya que en este modelo no podrá haber más de una transición por ciclo.
- No se consideran números enteros, y sólo para algunas operaciones se consideran vectores, por tanto, de manera general, el modelo los tendrá descompuestos en señales de un único bit. Las probabilidades y actividades sólo se tienen en cuenta a nivel de bits.
- Se ha elaborado una herramienta automática que elabora el modelo a partir de diseños descritos en VHDL. Ya que es una primera versión, la herramienta acepta un subconjunto del conjunto de VHDL para síntesis [60].
- Aunque el modelo considera las correlaciones espaciales dentro del circuito, las correlaciones espaciales a las entradas del circuito no son tenidas en cuenta.

En la última fila de la tabla 2-2 del capítulo anterior se han incluido las características generales de este modelo, de modo que sea fácilmente comparable con el resto de propuestas.

En comparación con el resto de modelos, las características que le faltan al modelo propuesto son la consideración de los **retardos**, las **correlaciones espaciales a las entradas** y los **elementos de memoria**. Se ha decido no incluirlas para poder construir un modelo más sencillo a fin de que sea más rápido de elaborar. Una vez comprobados los beneficios del modelo propuesto se podrán añadir estas tres características.

El no considerar todo el conjunto de VHDL para síntesis ni tampoco las señales agrupadas en vectores no supone una desventaja respecto al resto de propuestas debido a que son características propias del nivel RT y no del nivel de puertas.

Sin embargo, si se han tenido en cuenta señales agrupadas en vectores y algunos operadores de vectores para establecer un orden óptimo de los BDD.

3.3 Análisis estructural del circuito y modelo del hardware

En esta sección se explicará el análisis del circuito VHDL que conduce a la elaboración de un gráfico de dependencias de los nodos internos que se denominará **modelo del hardware**. Éste gráfico es **la estructura sobre la que sustenta el modelo probabilístico**, siendo **la base sobre la que se propagarán las probabilidades y actividades de los nodos del circuito**. Su elaboración es necesaria ya que por tratar circuitos en el nivel RT (y en concreto descritos en VHDL), no se dispone de una estructura que defina las influencias y dependencias, como las *netlists* disponibles en el nivel de puertas.

Este modelo del hardware acerca la descripción RTL al circuito en puertas, proporcionando una estructura similar al hardware que se sintetizará. El hecho de que haya muchos circuitos posibles resultantes de la síntesis constituye una dificultad y limitación añadida a la estimación en RTL, pues el circuito sintetizado final dependerá de la biblioteca tecnológica escogida, de las herramientas de síntesis empleadas y de las opciones y restricciones que se definan durante la síntesis. Por otro lado, el modelo del hardware propuesto es un circuito genérico, y por partir del nivel RT, no puede aspirar a la exacta definición proporcionada por el nivel de puertas.

Los pasos seguidos en el análisis se describirán en los siguientes apartados, que resumidamente son:

1. Elaboración de un modelo del hardware simplificado (SHM)
2. Extensión del modelo del hardware (EHM)

3. Identificación de la profundidad combinacional de las señales

Todas estas tareas se han automatizado en un programa informático, permitiendo mayor rapidez de análisis, y su uso en un amplio rango de diseños, especialmente para circuitos grandes, en los que la elaboración manual de estos modelos es inviable.

3.3.1 Modelo simplificado del hardware

El primer paso en la elaboración del modelo probabilístico es el *Modelo Simplificado del Hardware* (SHM[30]) [131]. Este modelo es elaborado automáticamente por Ardid [132], una herramienta de ayuda al diseño desarrollada en la División de Ingeniería Electrónica (UPM) y a la que se le añadirán los estudios de actividad desarrollados en esta tesis doctoral.

El SHM es un modelo orientado al hardware de las descripciones VHDL. En éste, las señales y sentencias VHDL están relacionadas según las dependencias e influencias que resultan de la implementación del circuito. Es un esquema a mitad de camino entre el *formato intermedio del VHDL* (VIF[31]) resultante de la compilación del diseño y el circuito sintetizado.

El modelo está diseñado de manera que pueda analizarse en capas según el nivel de detalle deseado. Existen tres niveles de detalle:

- Capa secuencial
- Capa combinacional
- Capa de sentencia

En cualquiera de los niveles existe acceso al código fuente VHDL. Esto permite relacionar los elementos del modelo con su correspondiente sentencia, expresión o elemento del diseño VHDL original.

La **capa secuencial** es la menos detallada, en ella, los elementos de memoria están claramente definidos pero no así la lógica combinacional. La lógica combinacional está vagamente especificada y podría asociarse con una nube difusa, en la que se conocen las entradas y salidas de las zonas combinacionales, pero no la lógica que las relaciona. De cada señal se pueden obtener sus **dependencias** e **influencias finales**[32]. En la figura 3-2 se muestra una representación esquemática de esta capa.

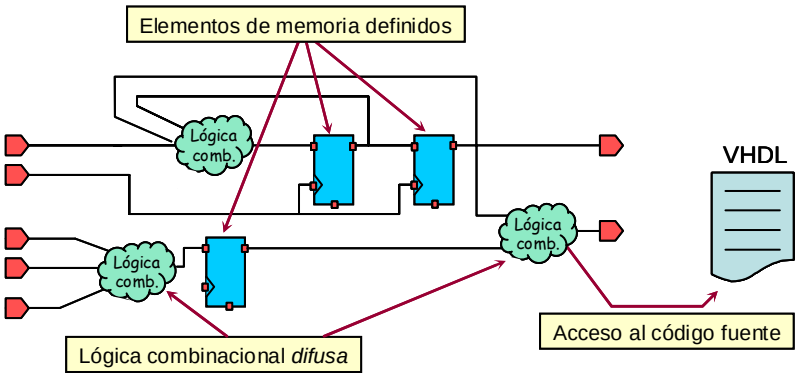


Figura 3-2: Representación esquemática del SHM en el nivel secuencial

El empleo del SHM permite la extracción de las zonas combinacionales del circuito, que están delimitadas por **nodos primarios**: puertos y elementos de memoria (figura 3-3), que constituyen las dependencias e influencias finales.

[30] Del inglés: *Simplified Hardware Model*
[31] Del inglés: *VHDL Intermediate Format*
[32] Se ha llamado *influencias finales de una señal X* a los puertos y elementos de memoria sobre los que la señal *X* influye a través de lógica combinacional. De manera similar, las *dependencias finales de una señal X*, son los puertos y elementos de memoria de los que la señal *X* depende mediante lógica combinacional.

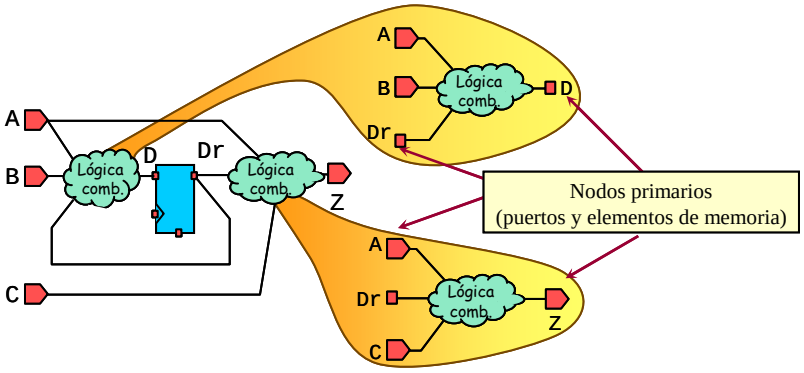


Figura 3-3: Extracción de las zonas combinacionales de un circuito

En la **capa combinacional** se pueden analizar las "nubes" combinacionales anteriormente descritas. Cada "nube" combinacional del circuito puede recorrerse interiormente, navegando por las dependencias e influencias de las señales a través de sentencias de asignación y procesos. Este nivel únicamente indica las dependencias entre señales, pero no la lógica que las relaciona.

La figura 3-4 muestra un diseño VHDL combinacional con sus capas secuencial (que en este caso no contiene elementos de memoria) y combinacional. En la capa combinacional, ahora son las sentencias y procesos los que están representados con una nube, indicando que su contenido está indefinido. En la figura, para facilitar la lectura, la capa combinacional se ha dibujado en dirección vertical, poniendo arriba las salidas (*Z*) y abajo las entradas (*A*, *B*, *C*, *D*, *E*, *F*). En medio, están los recuadros que representan las señales intermedias (*G*, *H*, *I*, *J*, *K*), conectados por las nubes combinacionales que simbolizan las sentencias.

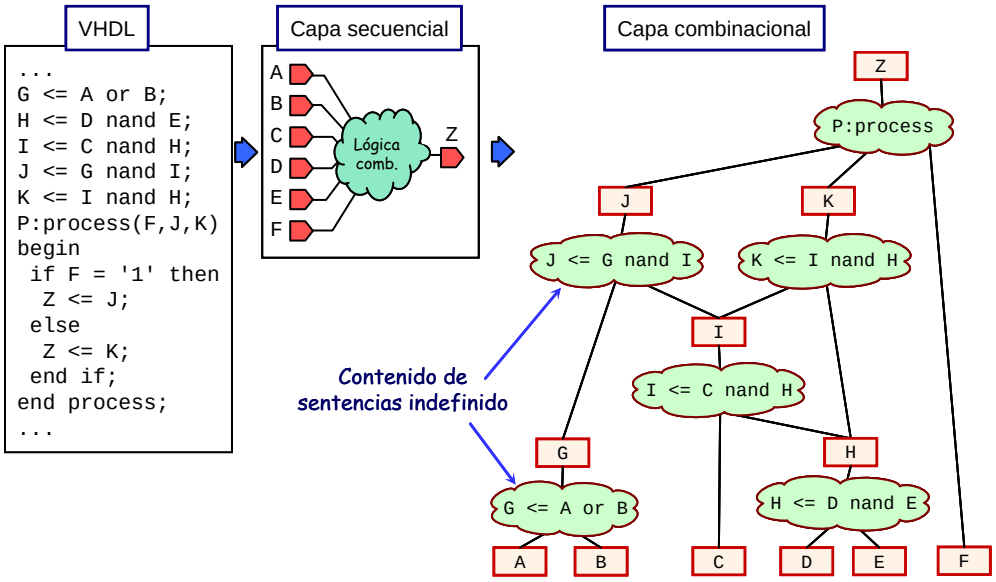


Figura 3-4: Capas secuencial y combinacional en el SHM para un diseño VHDL de ejemplo

En este nivel no se distingue entre tipos de sentencia y su complejidad. Una nube puede representar tanto una asignación simple como un proceso complejo. La única diferencia entre sentencias estriba en el número de entradas y salidas que tiene. Incluso, en la figura hubiese sido más apropiado representar las nubes sin indicar la sentencia a la que corresponde. Sólo se han incluido para facilitar la identificación en la lectura. Por el contrario, las señales, que están representadas mediante un recuadro, sí están definidas.

Por último, en la **capa de sentencia** pueden conocerse expresiones implicadas en cada sentencia o proceso. Este nivel no ofrece interpretación ni simplificación de las expresiones, simplemente permite el acceso a ellas tal cual fueron definidas en el código VHDL.

Para una de las sentencias del diseño de la figura 3-4, la figura 3-5 muestra de manera esquemática la diferencia de información de la capa combinacional respecto de la capa de sentencia. En la capa de sentencia, a partir de la señal asignada *G*, se llega a la sentencia de asignación (*G* <= *A* OR *B*), de ésta a la expresión que se asigna (*A* OR *B*), y por último al operador (OR) que relaciona los operandos que intervienen (*A*, *B*).

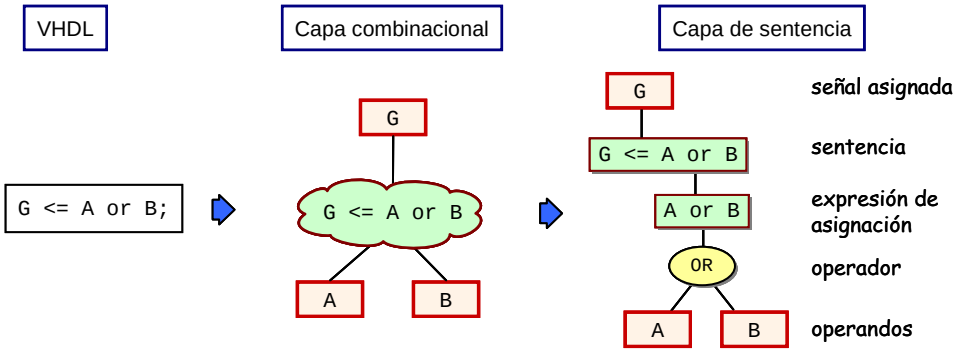


Figura 3-5: Capas combinacional y de sentencia para una asignación simple

En los procesos, la capa de sentencia es más compleja debido a que una señal puede recibir valores en más de una sentencia secuencial, y que además están afectadas por una o más condiciones[33]. La figura 3-6 muestra las capas combinacional y de sentencia para el proceso de la figura 3-4. En la representación, las sentencias se unen a la condición que les afecta por medio de un conector de trazo discontinuo, cuando esta condición afecta en su forma negada el conector termina con un punto negro.

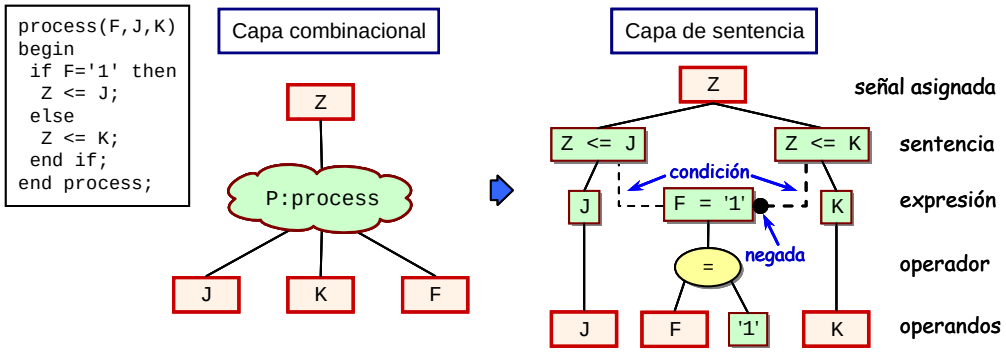


Figura 3-6: Capas combinacional y de sentencia para el proceso de la figura 3-4

Así, para el ejemplo de la figura 3-6, de la señal *Z* parten dos sentencias de asignación (*Z* <= *J* ; *Z* <= *K*), de cada una de estas asignaciones se llega por un lado a la expresión asignada (*J*, y *K* respectivamente) y, siguiendo la línea de trazo discontinuo se llega a la expresión de la condición. Obsérvese que la línea de trazo discontinuo de la derecha termina en un punto negro que simboliza que la sentencia de asignación (*Z* <= *K*) está afectada negando la condición (*F* = '1'). Por último, de la misma manera que en el ejemplo anterior, de las expresiones se llega a los operadores si los hubiese, y de éstos, a los operandos.

El ejemplo anterior es muy elemental, en la figura 3-7 se muestra un ejemplo de mayor complejidad. En este ejemplo, la señal *Z* está asignada en tres sentencias, y por tanto, del recuadro cuelgan tres asignaciones. Como hay dos sentencias condicionales, las líneas discontinuas muestran cómo afectan estas sentencias condicionales a las sentencias de asignación. Así, la condición (*A*='1') afecta a las sentencias (*Z* <= NOT *E*) y (*Z* <= *D*) y afecta de forma negada a la sentencia (*Z* <= *E* XOR *F*). De la misma manera que en los otros ejemplos, a partir de las expresiones de cada sentencia se llega a los operadores y de éstos a los operandos. Obsérvese cómo cuando hay expresiones complejas, la expresión se separa en operandos según su prioridad. Por tanto, de la expresión [(*B* OR *C*) = '1'] se llega primero al operador igual (=) y

[33] Las sentencias concurrentes del VHDL también pueden estar afectadas por condiciones, y su tratamiento sería similar al de un proceso.

de éste cuelgan por un lado el operador OR y por otro lado la constante '1'. Del operador OR cuelgan los operandos (*B*, *C*).

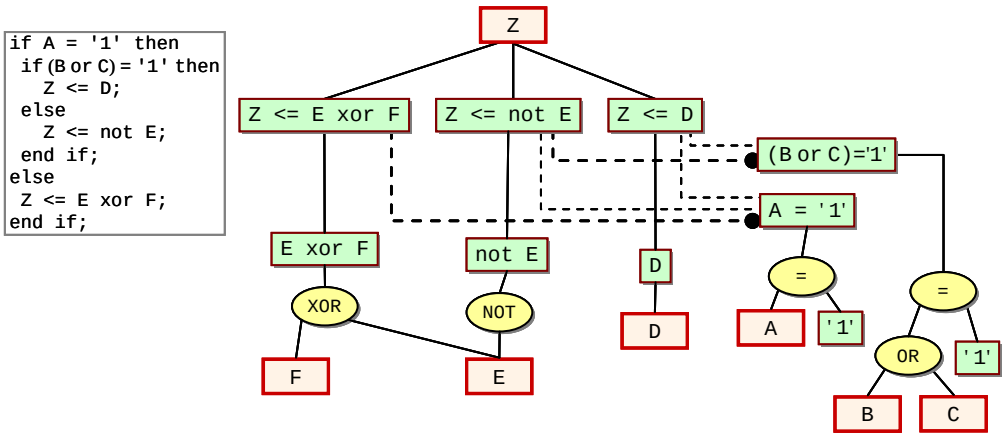


Figura 3-7: Capa de sentencia para un proceso ejemplo

3.3.2 Modelo extendido del hardware

El modelo simplificado (SHM) resulta muy útil porque permite un cómodo acceso a las influencias y dependencias de las señales. Para cada señal, el uso de la capa secuencial permite el acceso a sus influencias y dependencias finales, y por otro lado, si se quiere conocer las influencias y dependencias inmediatas se recurre a la capa combinacional.

Sin embargo, **la capa de sentencia no ofrece una información que sea fácilmente manejable**, precisándose de un análisis para interpretar la función lógica que representa. En particular, las sentencias condicionales no están adecuadamente procesadas para facilitar su interpretación, por ejemplo, en la figura 3-6 se muestra la capa de sentencia para un ejemplo muy sencillo, en donde solamente hay una condición.

En un proceso más complejo como el de la figura 3-7 se puede apreciar cómo de esta capa no resulta fácil extraer la jerarquía de las condiciones ni de las asignaciones.

Debido a que las sentencias condicionales son muy habituales en VHDL (implementándose por ejemplo mediante sentencias *if* y *case*, y con sentencias concurrentes condicionadas), se precisa de una extensión del modelo simplificado en la que se mejore su representación. Este modelo extendido se ha llamado *modelo extendido del hardware*, que para simplificar se nombrará con las siglas en inglés EHM (*Extended Hardware Model*).

La extensión que se propone se muestra en la figura 3-8, en la que se ha puesto el mismo proceso VHDL que en la figura 3-4, facilitando así la comparación de esta extensión con la de la capa de sentencia del SHM.

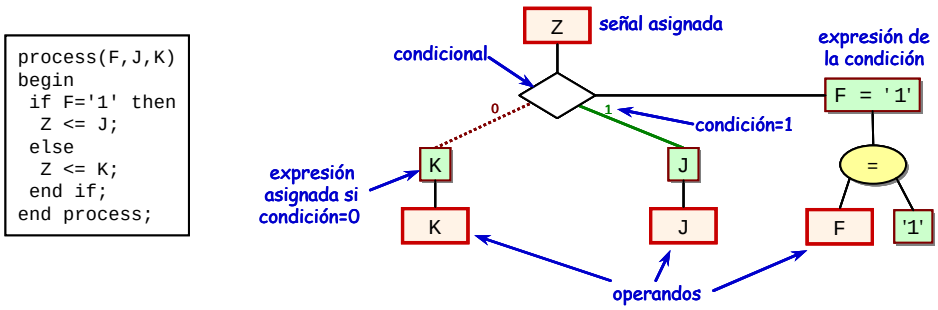


Figura 3-8: EHM en sentencias condicionales, proceso tomado del ejemplo de la figura 3-6

En esta extensión del modelo, a partir la señal asignada (Z) se accede a la condición que afecta a las sentencias de asignación. De la condición salen tres ramas, una horizontal y dos inclinadas:

- La horizontal apunta a la expresión de la condición (*F* = '1').
- La rama inclinada con trazo punteado lleva a la expresión (*K*) que se asignará a la señal (*Z*) en caso de que no se cumpla la condición (*F* = '1').
- La rama inclinada con de trazo continuo apunta a la expresión (*J*) que se asignará en caso de que se cumpla la condición (*F* = '1').

Por último, de cada expresión se llega a las señales que intervienen (*K*, *J*, *F*).

Para un caso más complejo como el ejemplo de la figura 3-7 la extensión del modelo se muestra en la figura 3-9. Como se puede apreciar, de la extensión del modelo se extrae fácilmente la jerarquía de las condiciones y asignaciones. En donde la condición más externa (*A*='1') cuelga directamente de la señal asignada (*Z*). El cumplimiento o no de esta condición determinará qué rama seguir:

- Si se cumple la condición (*A*='1') se llegará a la segunda condición [(*B* OR *C*)='1']
- Si no se cumple la condición, se llegará a la asignación (*E* XOR *F*).

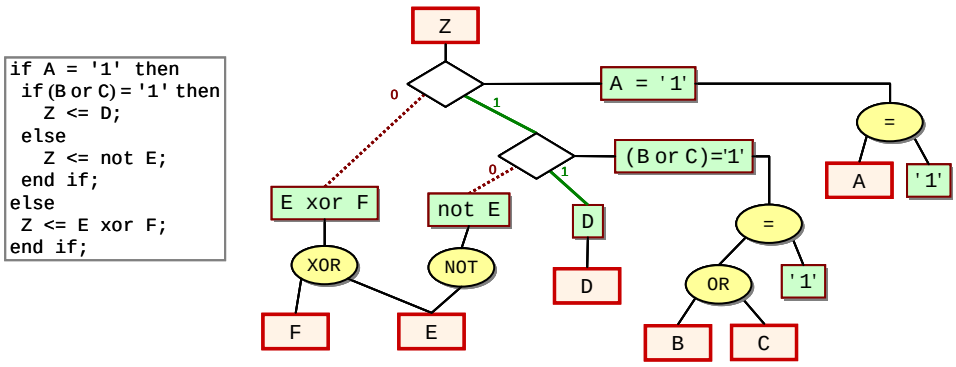


Figura 3-9: Extensión del modelo para el ejemplo de la figura 3-7

Para terminar, se debe recalcar que la estructura propuesta se asemeja más al hardware que se implementará, en la que las condiciones del modelo extendido se pueden asimilar a multiplexores. La figura 3-10 muestra esta similitud, en donde para simplificar la figura no se han desarrollado las expresiones. Por tanto, **el modelo extendido del hardware es una aproximación al hardware que se sintetizará, proporcionando una estructura más eficiente para el análisis del circuito.**

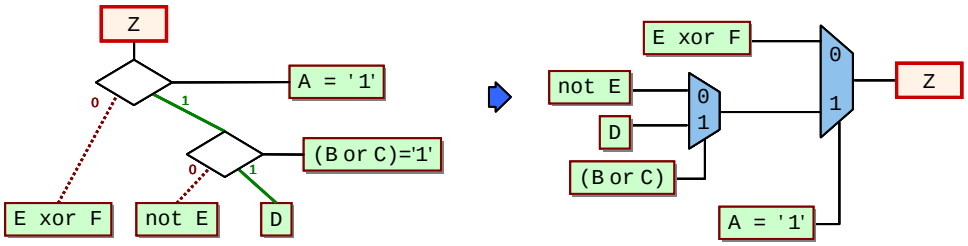


Figura 3-10: Equivalencia del EHM con el hardware

3.3.3 Profundidad combinacional

Una vez que se dispone del modelo del hardware con las dependencias e influencias de cada señal, se establece una jerarquía entre las señales según sus relaciones de dependencia que determine un orden para el análisis probabilístico. Siguiendo este orden jerárquico, se asegura que antes de que se analizase cualquier señal se hayan analizado todas sus dependencias. Este orden se constituye para cada una de las zonas combinacionales del circuito (figura 3-3), esto es, en la capa combinacional del SHM.

Para establecer esta jerarquía, a todos los nodos internos de un circuito combinacional (o los de cada una de las partes combinacionales de uno secuencial) se les asigna un valor de *profundidad combinacional* (CD[34]) que indica la distancia de su dependencia final más lejana.

Las dependencias finales (son nodos primarios) no tienen profundidad combinacional (CD = 0), el resto de señales tienen un valor superior a cero. Para calcular la profundidad combinacional (CD_X) de una señal cualquiera X, se le añade uno a la profundidad combinacional de su dependencia inmediata con mayor profundidad combinacional.

$$CD_X = MAX_{i \in Inm_X} \{CD_i\} + 1 \quad (3.1)$$

Donde Inm_X es el conjunto de dependencias inmediatas de la señal X.

La profundidad combinacional proporciona un medio para ordenar los nodos y así realizar el análisis empezando por las señales con más baja profundidad combinacional, de tal manera que no se realiza el análisis de un nodo sin que todas sus dependencias hayan sido analizadas.

En la figura 3-11 se muestra el resultado de calcular la profundidad combinacional para el diseño VHDL del ejemplo de la figura 3-4.

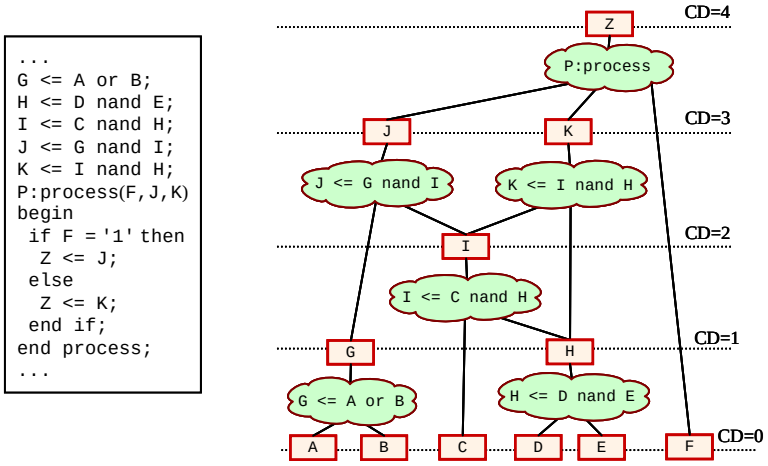


Figura 3-11: Profundidad combinacional para el diseño de la figura 3-4

Para representarlo de una manera más familiar, en la figura 3-12 se muestra el mismo diseño en RTL, en donde también se indica la profundidad combinacional de cada una de los nodos del circuito.

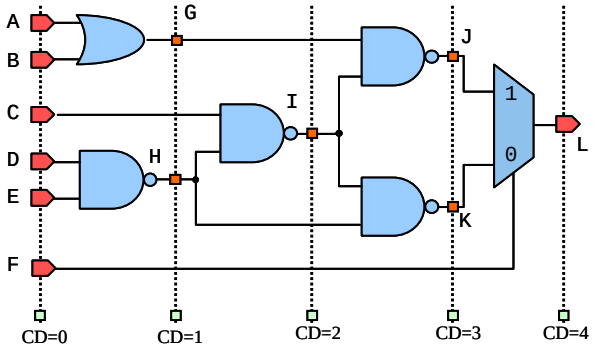


Figura 3-12: Profundidad combinacional para el diseño de la figura 3-4 representado en RTL

Durante el análisis del circuito se estudian primero las señales con menor profundidad combinacional. Así, por ejemplo, las señales G y H se analizan antes que la señal I, y ésta antes de J y K, la última que se analizará será la señal L. Este tipo de análisis ha sido empleado también por otros autores para el análisis del consumo [98], [23], llamándolo *nivelación* (*levelization*).

[34] Del inglés: *Combinational Depth*

3.4 Partición del circuito

Con el modelo del hardware (EHM) que se acaba de explicar se dispone de la estructura que permite elaborar las funciones de probabilidad y actividad de los nodos del circuito. A partir de estas funciones y por medio del modelo del hardware se pueden propagar las probabilidades y actividades de señal desde las entradas a las salidas.

Sin embargo, como ya se ha comentado, los **modelos probabilísticos tienden a crecer excesivamente con el tamaño de los circuitos**. Por ello **es necesario dividir el modelo** para limitar el uso de recursos computacionales y de memoria. No obstante, para que el modelo sea exacto, la partición se debe realizar **teniendo en cuenta las dependencias estructurales** del circuito (§2.2.3.1).

El análisis de las dependencias estructurales lleva a la partición del circuito en *regiones de reconvergencia*, también llamadas superpuertas (véase la figura 2-20). En circuitos grandes y con muchas dependencias estructurales el tamaño de estas particiones todavía puede resultar excesivo, haciendo necesario aproximar el método mediante la limitación de la profundidad de estas regiones (véase la figura 2-21).

En consecuencia, además de la partición en regiones reconvergentes, en esta tesis se propone la división del circuito en *regiones disjuntas* debido a que permiten dividir las regiones de reconvergencia en áreas más pequeñas.

En los siguientes apartados se explicará:

- La identificación de regiones disjuntas en el nivel RT
- El cálculo de la actividad en regiones disjuntas
- El error en el cálculo de la actividad por el uso de regiones disjuntas

Además, en el anexo II se amplían ciertos aspectos relativos a las particiones. En el apartado AII.1 se explican los algoritmos desarrollados para la partición del modelo del hardware. En el apartado AII.2 se amplía la información sobre el error cometido en el cálculo de la actividad por el uso de regiones disjuntas.

3.4.1 Identificación de regiones disjuntas

Como se mostró en el apartado 2.3, la mayoría de las propuestas basan la partición en la búsqueda de señales independientes. La propuesta de Agrawal ([2] página 37) hace uso de señales **mutuamente disjuntas** para dividir el circuito en particiones más pequeñas. Este método se aplica en el nivel de puertas, y para identificar las señales disjuntas, el algoritmo realiza un análisis de contradicción, detectando las señales disjuntas mediante un proceso similar a la simulación.

Sin embargo, desde el nivel RT se puede identificar fácilmente la existencia de muchas de las señales disjuntas de un circuito sin tener que recurrir a dichos algoritmos de detección. Las sentencias condicionales, en concreto las sentencias *if* y *case* en VHDL, pueden hacer que nodos internos del circuito sean mutuamente disjuntos. Esto hace que su identificación sea inmediata.

En la mayoría de los casos, una sentencia condicional se convierte en un multiplexor en el que la condición selecciona la entrada que se asigna a la señal. En la sección siguiente se verá cómo los multiplexores crean señales disjuntas. La figura 3-13 muestra distintas representaciones de un multiplexor: en VHDL, en el nivel RT, en el modelo extendido del hardware (§ 3.3.2) y en una de las representaciones posibles en puertas lógicas.

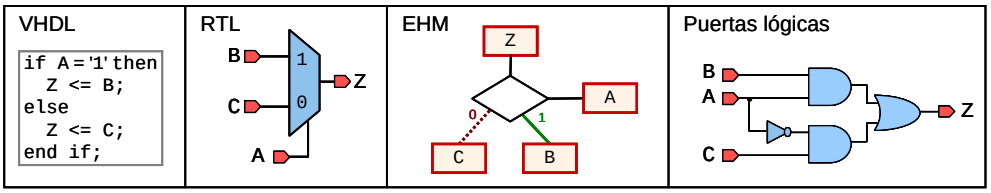


Figura 3-13: Distintas representaciones para un multiplexor

La identificación de estas estructuras en RTL o en el EHM es inmediata, solamente hay que buscar el multiplexor en el RTL o la condición (rombo) en el EHM. Sin embargo, en el nivel de puertas no lo es tanto, ya que o bien puede tomar otra disposición o bien algunas puertas se han podido simplificar o agrupar con otras cercanas.

A continuación se justifica teóricamente por qué un multiplexor produce señales disjuntas que permiten dividir el circuito, y en el siguiente subapartado se mostrarán algunos sencillos ejemplos donde se evidencian los beneficios de esta propuesta.

3.4.1.1 Señales disjuntas en multiplexores

Un multiplexor produce nodos cuyos eventos nunca podrán darse simultáneamente. Esto es por definición, ya que un multiplexor selecciona una y sólo una de las entradas para asignarla a la salida. Por tanto, dentro de un multiplexor se generan señales mutuamente disjuntas.

De la misma manera, una sentencia condicional produce eventos mutuamente disjuntos, ya que las alternativas *else* o *elsif* dentro de una sentencia *if*, y las alternativas de la sentencia *case* son mutuamente exclusivas para el lenguaje VHDL [59].

Así, como se muestra en la figura 3-14, la sentencia *if* produce eventos mutuamente disjuntos que se han llamado *D* y *E*. Estos eventos no se pueden dar a la vez, ya que, no puede pasar que *A*='1' y a la vez *A*='0'.

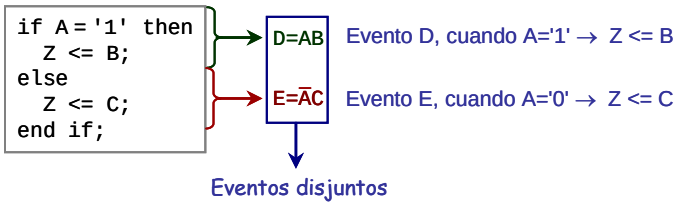


Figura 3-14: Generación de nodos disjuntos en una sentencia condicional (multiplexor)

Debido a que son eventos disjuntos, la función de probabilidad de *Z* se puede expresar en función de *D* y *E* (figura 3-15). Este paso no implica ningún error en el cálculo.

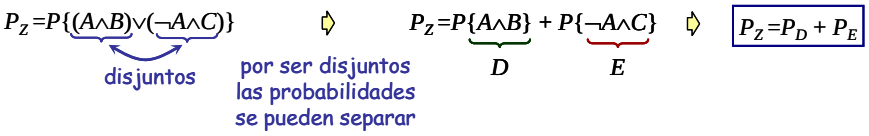


Figura 3-15: Separación de las probabilidades de los eventos D y E

En la figura 3-16 se muestra dónde estarían los nodos *D* y *E* en la representación en puertas.

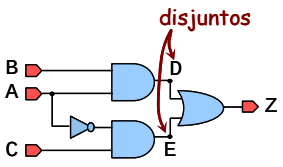


Figura 3-16: Localización de los nodos D y E en el circuito en puertas

Es importante señalar que el cálculo puede realizarse de este modo **incluso cuando las señales de entrada al multiplexor no sean independientes**, y que no se trata de una aproximación, esto

es, el cálculo de la probabilidad llevado a cabo mediante este procedimiento es exacto y es equivalente al realizado sin considerar las señales disjuntas. Por tanto, **para el cálculo de la probabilidad de la señal de salida de un multiplexor no se necesita realizar el análisis de reconvergencia**.

En los circuitos electrónicos digitales es frecuente que el camino de control y el camino de datos sean independientes. En tal caso, se puede poner la probabilidad de la salida directamente en función de las entradas, en vez de ponerla en función de unos nodos internos del multiplexor que no existen en el código VHDL ni aparecen en la representación RTL (como lo serían los *D* y *E* de la figura 3-14). Haciéndolo de esta manera se facilita la representación de los ejemplos, por lo que en la mayoría de los ejemplos que aparecen en esta tesis se mostrarán de esta forma.

La figura 3-17 muestra este caso, en donde las alternativas del multiplexor (*B*, *C*) pueden tener fuentes comunes, pero son independientes respecto a la condición (*A*). Para este caso, el cálculo de la probabilidad sigue siendo sencillo.

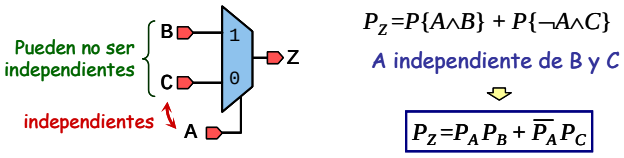


Figura 3-17: Cálculo de la probabilidad con la condición independiente

3.4.1.2 Ejemplos de aplicación de las regiones disjuntas

En este apartado se mostrará con unos sencillos ejemplos las ventajas de la partición del circuito usando las señales disjuntas que se generan en las sentencias condicionales.

En la figura 3-18 se muestra un ejemplo en donde hay reconvergencia, provocando que ambas alternativas del multiplexor dependan de la señal *C*. Esto hace que las señales *H* y *I* sean dependientes, y por tanto, si se realizase el cálculo de la probabilidad de *Z* bajo la condición de independencia, habría que tomar la región de reconvergencia. Como se puede ver en el dibujo central de la figura, esta región abarca la totalidad del circuito, y es la región que tomarían la mayoría de las propuestas; solamente Agrawal [2], en el nivel de puertas, tomaría la región disjunta (dibujo de la derecha) en caso de que su algoritmo la detectase, pues su algoritmo no detecta todas las señales disjuntas de un circuito. Sin embargo, en el nivel RT la detección de estas zonas disjuntas es inmediata y prácticamente carece de coste computacional.

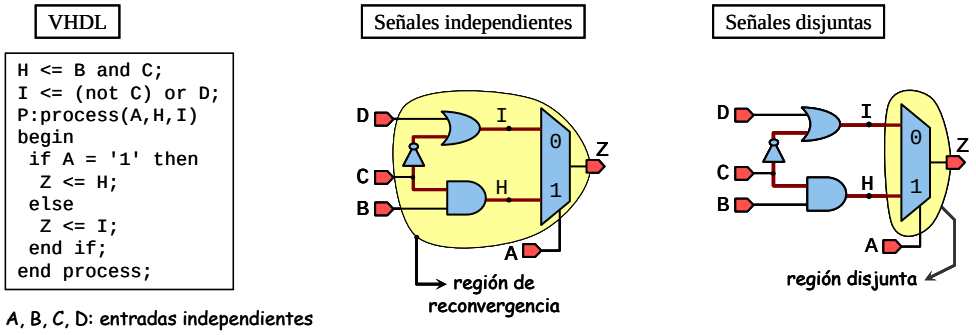


Figura 3-18: Regiones para el cálculo de la probabilidad de Z considerando señales independientes (centro) y las señales disjuntas producidas en un multiplexor (derecha)

En un circuito algo más complejo como el de la figura 3-4 los beneficios de esta partición son más notorios ya que la región de reconvergencia abarca más área. En el dibujo de la izquierda de la figura 3-19 se ve cómo ambas alternativas del multiplexor dependen de las señales *H* e *I*, la región de reconvergencia tiene que cubrir ambas señales. Sin embargo, en la figura de la derecha se muestra cómo la región disjunta cubre tan sólo las entradas del multiplexor.

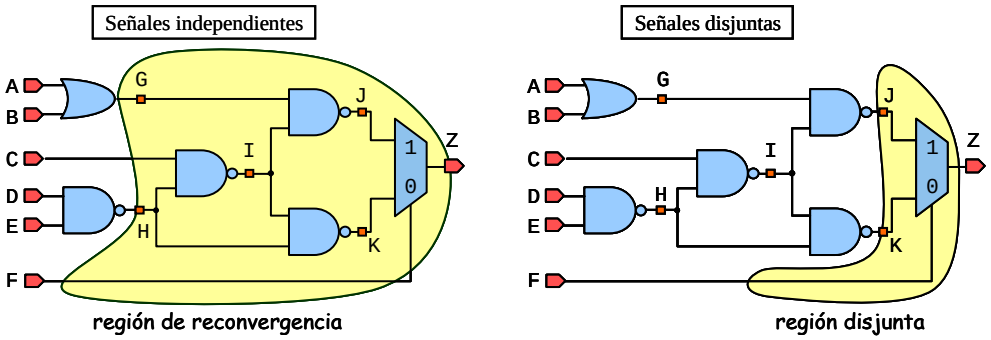


Figura 3-19: Regiones de reconvergencia y disjunta para el ejemplo de la figura 3-4

Por último, la figura 3-20 muestra la aplicación del método en sentencias condicionales anidadas (es el mismo ejemplo de la figura 3-9). Con el fin de facilitar la interpretación de la figura, se han marcado las zonas del código VHDL que representan los nodos internos generados (*K*, *L* y *M*). Las sentencias condicionales anidadas se pueden sustituir por un nodo interno, como lo hace el nodo *K* que representa la sentencia condicional interna.

La región disjunta para la señal *Z* es la correspondiente a la sentencia condicional externa. Por el contrario, la región de reconvergencia es mayor debido a que tiene que incluir a la señal *E* por su influencia en ambas alternativas del multiplexor externo.

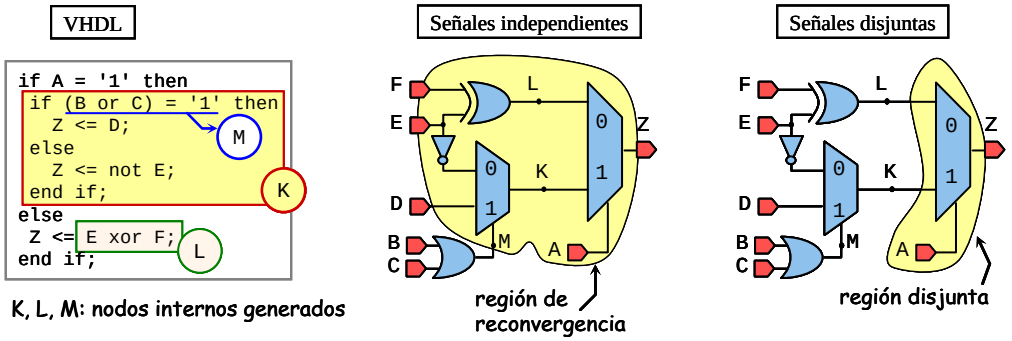


Figura 3-20: Regiones de reconvergencia y disjunta en un proceso con sentencias condicionales anidadas

La existencia de sentencias condicionales anidadas no supone ninguna dificultad adicional en el procesamiento, y son equivalentes al diseño con el proceso descompuesto en procesos y asignaciones concurrentes. Para el ejemplo tratado, la figura 3-21 muestra cómo sería esta descomposición en el VHDL, en donde se han creado las señales *K* y *L*, y la asignación de éstas se hace en un proceso y una sentencia concurrente externos al proceso donde dan valor a *Z*.

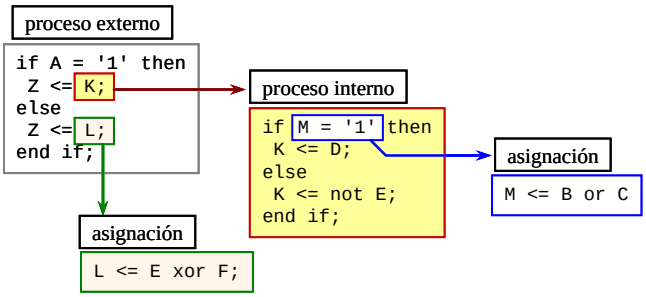


Figura 3-21: Descomposición del proceso con sentencias condicionales anidadas en sentencias concurrentes

3.4.2 Cálculo de la actividad en regiones disjuntas

Se ha visto que la partición del circuito en regiones disjuntas provenientes de sentencias condicionales proporciona un medio eficaz para la simplificación del cálculo de la probabilidad

de señal. Esta simplificación del cálculo no provoca pérdida de exactitud para la probabilidad de señal, obteniéndose el mismo resultado con partición que sin ella. Sin embargo, no ocurre lo mismo para el cálculo de la actividad.

La ecuación de la actividad de un multiplexor simple como el de la figura 3-17 se puede construir mediante la expansión de *Shannon* para la actividad (ecuación 2.30), extrayendo en este caso los cofactores $a(Z)_{A_{i\to j}}$ de la señal de selección *A*.

$$a_Z = P(A_{1\to1})\cdot a(Z)_{A_{1\to1}} + P(A_{0\to0})\cdot a(Z)_{A_{0\to0}} + \\ + P(A_{0\to1})\cdot a(Z)_{A_{0\to1}} + P(A_{1\to0})\cdot a(Z)_{A_{1\to0}} \qquad (3.2)$$

Al realizar la partición disjunta, los cofactores $a(Z)_{A_{0\to1}}$, $a(Z)_{A_{1\to0}}$ dependerán exclusivamente de las alternativas *B* y *C*, y no de la señal de selección *A*. Pero cuando las señales *B* y *C* no son independientes, para no cometer error, el cálculo del valor de los cofactores no se debería realizar en función de dichas señales, sino en función de fuentes independientes.

Para explicar este concepto se utilizará como ejemplo un multiplexor como el de la figura 3-22. En él, la señal de selección *A* es independiente de las alternativas, la señal de selección es la señal que provoca que las alternativas se conviertan en disjuntas. Por otro lado, las alternativas *B* y *C* tienen una dependencia común *D*, por lo tanto no son independientes entre sí, pero sí lo son respecto de *A*.

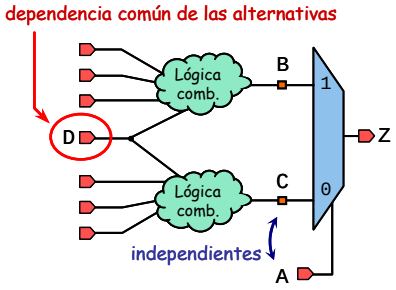


Figura 3-22: Multiplexor con alternativas dependientes de la señal D

Cuando la señal de selección *A* permanece invariable durante dos ciclos de reloj (*t*=0, *t*=*T*), la salida *Z* recibe el valor de la misma alternativa (*B* ó *C*) en los dos ciclos y, por tanto, la actividad de *Z* será la misma que la de la alternativa seleccionada.

Para mostrarlo en un ejemplo, en la figura 3-23 aparece el caso en donde la señal *A* se mantiene a cero durante dos ciclos de reloj consecutivos. Bajo esta condición, el circuito se podría sustituir por el de la derecha de la figura, en donde la salida *Z* recibe directamente el valor de *C*. Por lo tanto, la actividad de la salida *Z* será la misma que la de la alternativa *C*, sin que afecte la dependencia común *D* de ambas alternativas. Éste es el caso del cofactor $a(Z)_{A_{0\to0}}$.

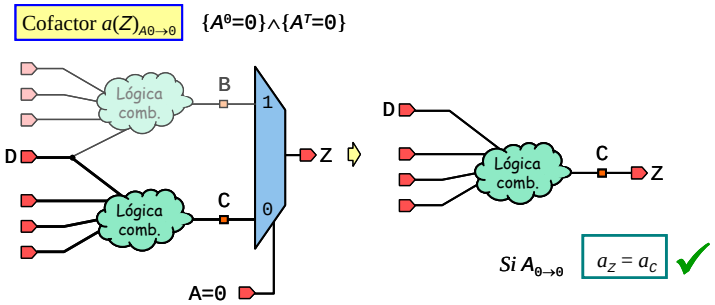


Figura 3-23: Actividad de Z cuando la selección se mantiene en cero: cofactor $a(Z)_{A_{0\to0}}$

De la misma manera, para el caso en que la señal de selección *A* permanezca con valor 1 durante dos ciclos de reloj consecutivos, la actividad de la salida *Z* sería igual a la actividad de la alternativa *B*. Y sería el caso del cofactor $a(Z)_{A_{1\to1}}$.

Cuando existe una transición en la señal de selección se pasa de asignar una alternativa a la otra. En el caso en que las alternativas sean independientes el valor de una alternativa no

influye en el valor de la otra alternativa en el intervalo siguiente. Pero si tienen una dependencia común, al considerar dependencias temporales, si puede existir influencia.

A continuación y a través de la figura 3-24 se muestra cómo se produce este efecto:

- La señal D influye en las alternativas B y C
- Por la correlación temporal, la señal D en tiempo T (D^T) está condicionada por el valor que tenía en el tiempo 0 (D^0)
- Si hay transición de 0 a 1 en la señal de selección ($A_{0\to1}$), entonces:
  - En $t = 0$ la alternativa C^0 es asignada a la salida Z^0
  - En $t = T$ la alternativa B^T es asignada a la salida Z^T
- Por tanto, la actividad de Z (esto es, a_Z) depende de las alternativas C^0 y B^T
- En consecuencia, no es exacto calcular la actividad de Z directamente a partir de los valores de probabilidad de las alternativas C^0 y B^T, ya que éstas tienen a D^0 como dependencia común:
  - C^0 depende de D^0
  - B^T depende de D^T; D^T está condicionada por D^0

Así que realizar la partición disjunta en un multiplexor cuando las alternativas son mutuamente dependientes puede llevar a resultados de actividad aproximados, ya que la partición hace que los cofactores $a(Z)_{A_{0\to1}}$ y $a(Z)_{A_{1\to0}}$ puedan no ser exactos.

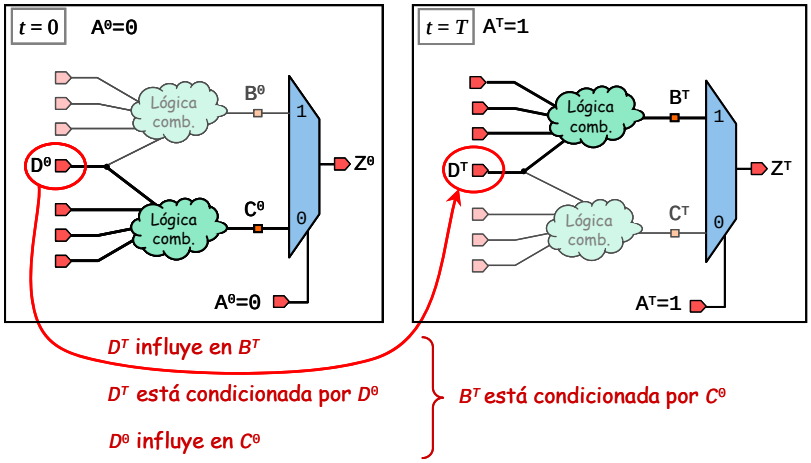


Figura 3-24: Influencia de una dependencia común al cambiar la selección de las alternativas de un multiplexor

Ahora se mostrarán las ecuaciones que resultan de realizar el cálculo mediante regiones disjuntas.

La ecuación del cofactor $a(Z)_{A_{0\to1}}$ del circuito de la figura 3-24 es:

$$a(Z)_{A_{0\to1}} = P\{(C^0=1)\wedge(B^T=0)\} + P\{(C^0=0)\wedge(B^T=1)\} \tag{3.3}$$

Si no se considera la dependencia común D, las probabilidades se pueden descomponer, quedando la ecuación aproximada:

$$a(Z)_{A_{0\to1}} \approx P\{C^0=1\}\cdot P\{B^T=0\} + P\{C^0=0\}\cdot P\{B^T=1\} \tag{3.4}$$

Que por ser estrictamente estacionario (ecuación 2.2):

$$a(Z)_{A_{0\to1}} \approx a'(Z)_{A_{0\to1}} = P_C \cdot \overline{P_B} + \overline{P_C} \cdot P_B \tag{3.5}$$

Cuando sea necesaria la distinción, se añadirá un apóstrofo (') a las actividades relativas al cálculo aproximado para diferenciarlas de las del cálculo exacto, así a'_Z ó $a'(Z)$ representará la actividad de Z aproximada por el uso de regiones disjuntas.

Para resumir las ecuaciones de los cofactores y sus aproximaciones, la figura 3-25 muestra los cuatro cofactores $a(Z)_{A_{i\to j}}$. Los dos de arriba están relacionados con la inactividad de *A*, y por ello se denominarán *cofactores de inactividad*; mientras que los dos de abajo están relacionados con la actividad de *A*, y por tanto se denominarán *cofactores de actividad*.

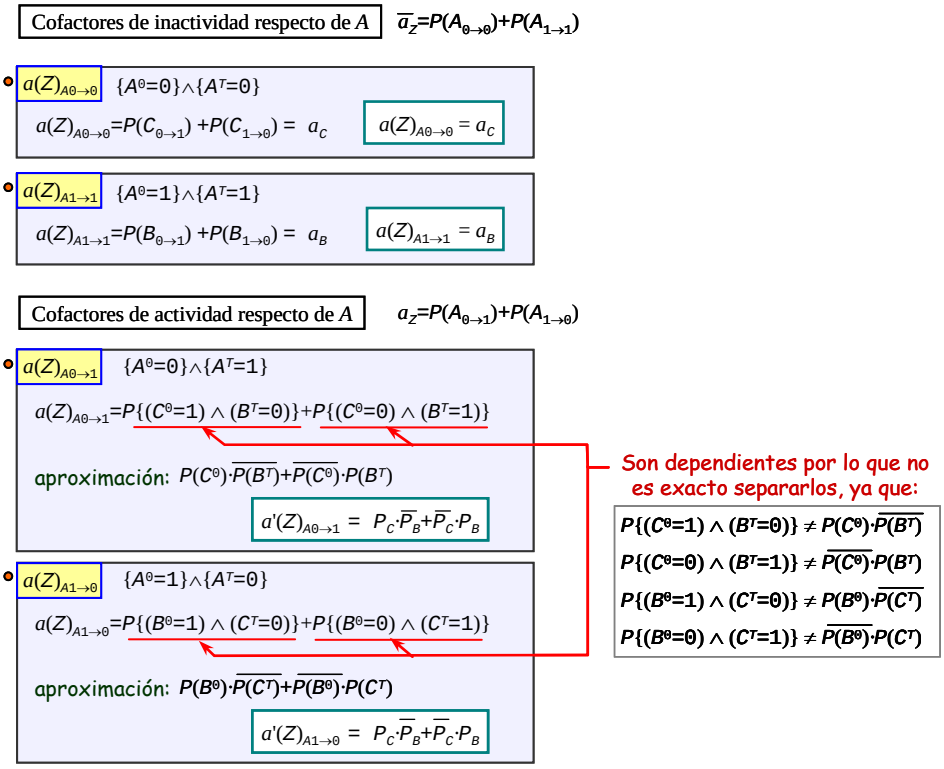


Figura 3-25: Los cuatro cofactores $a(Z)_{A_{i\to j}}$ del multiplexor de la figura 3-22

Y así, tomando la ecuación 3.2 y las de la figura 3-25, la ecuación aproximada de la actividad queda:

$$a_Z \approx a'_Z = P(A_{1\to1})\cdot a_B + P(A_{0\to0})\cdot a_C +$$
$$+ P(A_{0\to1})\cdot [P_C\cdot\overline{P}_B + \overline{P}_C\cdot P_B] + P(A_{1\to0})\cdot [P_C\cdot\overline{P}_B + \overline{P}_C\cdot P_B]$$

Que por la ecuación 2.5 resulta:

$$a_Z \approx a'_Z = P(A_{1\to1})\cdot a_B + P(A_{0\to0})\cdot a_C + a_A\cdot [P_C\cdot\overline{P}_B + \overline{P}_C\cdot P_B] \quad (3.6)$$

En la que los dos primeros sumandos son exactos y el último es aproximado. Se debe observar la sencillez de la fórmula resultante, que depende únicamente de las entradas al multiplexor, evitando con ello buscar las señales independientes y el uso de regiones de reconvergencia muy grandes. En el siguiente apartado se analizará el error cometido en esta aproximación para conocer su magnitud y saber si es aceptable.

3.4.3 Análisis del error de la actividad en regiones disjuntas

De la ecuación 3.6 se puede observar que cuanto menor sea la actividad de la señal de selección, a_A, menor es la influencia de la aproximación. Esto se debe a que el último sumando es el aproximado y es el que está relacionado con la actividad de *A*.

La actividad e inactividad de una señal suman la unidad (ecuación 2.6), por tanto el error se acentúa aún más para actividades altas de la señal de selección, y al contrario, se minimiza con actividades bajas.

Además, en el apartado 3.4.2 (y el anexo AII.2) se ha mostrado que para la actividad su cálculo es aproximado cuando existen reconvergencias que afectan a las entradas del multiplexor. Con

esto y a partir de la ecuación 3.6 se pueden extraer unas condiciones que hacen minimizar el error cometido (véase el anexo AII.2):

1. La señal de selección es independiente de las alternativas
2. La actividad de la señal de selección es baja
3. Existe una baja dependencia temporal en las señales reconvergentes
4. La relación entre las fuentes comunes de las alternativas es pequeña respecto a las independientes

Estas cuatro condiciones se han representado en la figura 3-26.

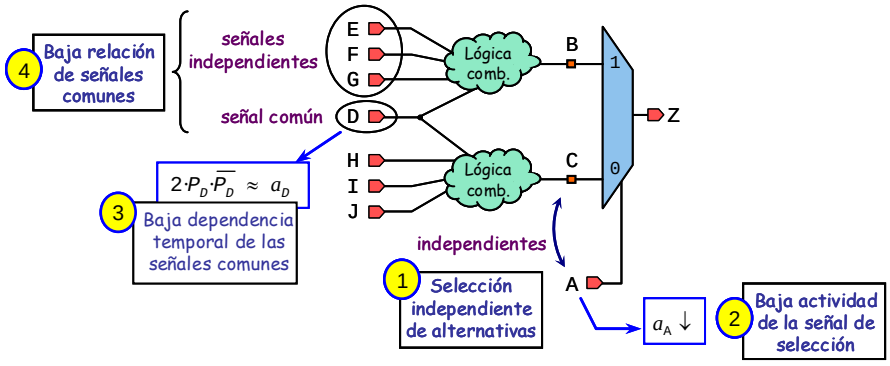


Figura 3-26: Condiciones que minimizan el error cometido en el cálculo de la actividad mediante regiones disjuntas

A continuación se explicará el fundamento de estas condiciones. En el anexo AII.2 se analizan los errores de forma práctica con circuitos de ejemplo. La consulta de estos ejemplos podría facilitar la comprensión de lo que a continuación se expone.

No es necesario que se cumplan la totalidad de las condiciones anteriores, aunque sí se ha estimado imprescindible que **se cumpla la primera condición**: que la señal de selección sea independiente de las alternativas. Con esto sólo se emplean las regiones disjuntas en aquellos circuitos en los que la señal de selección se pueda separar de las alternativas. Esto sucede en muchos circuitos, en los que el camino de control (selección) es independiente del camino de datos (alternativas).

El cumplimiento de esta condición hace que sólo los cofactores de actividad (§3.4.2, ecuación 3.5) sean aproximados, mientras que los dos primeros sumandos de la ecuación de actividad sean exactos (ecuación 3.6). El cumplimiento de esta condición justifica la siguiente.

La segunda condición impone una baja actividad de la señal de selección. Esto hace disminuir la influencia del cofactor de actividad, y por tanto el error cometido (ecuación 3.6). Esto es así siempre y cuando se cumpla la primera condición: que la señal de selección sea independiente de las alternativas.

Si se cumple la **tercera condición** el error se minimiza al haber una baja dependencia temporal de las señales comunes. Para el caso extremo en que no exista dependencia temporal (ecuación 2.21) no se cometerá error. Nótese que todas las simplificaciones de las ecuaciones de la sección 3.4.2 que resultan en la ecuación 3.6 se basan en esta aproximación: la actividad de la señal común en el cofactor de actividad se aproxima a su ecuación de independencia temporal.

La **cuarta condición** indica que hay una baja proporción de señales comunes afectando a las alternativas. Consecuentemente la influencia de la reconvergencia es pequeña, pues son las actividades de estas señales las únicas que se aproximan. Esto se muestra en el anexo AII.2, en donde se pueden comparar los ejemplos de la figura AII-23 y AII-24. Ambos circuitos tienen una estructura similar, sin embargo el error cometido en el primero es claramente mayor debido a que las alternativas comparten las mismas dependencias.

De las cuatro condiciones, la primera y la cuarta son condiciones que dependen de la estructura del circuito, y por tanto no varían. En consecuencia, se denominarán **condiciones estructurales**

o intrínsecas. Las condiciones segunda y tercera dependen de los valores de probabilidad y actividad de las señales implicadas, y por tanto podrán variar según las circunstancias en las que está el circuito. Consecuentemente, se denominarán **condiciones circunstanciales o extrínsecas**.

Por tanto, a la hora de realizar la partición del circuito se debe de analizar el cumplimiento de estas condiciones. Durante el proceso de partición, **gracias al EHM** (§3.3.2) **se puede obtener la información necesaria para evaluar tanto las condiciones estructurales como circunstanciales**. En el capítulo de resultados experimentales (capítulo 4) se verá que normalmente en un circuito se pueden realizar múltiples particiones disjuntas. Sin embargo, como las particiones introducen error en el cálculo, éstas no deben aplicarse de manera exhaustiva, sino que se deben aplicar en aquellos puntos que por su estructura y condiciones, se minimice el error, y se consiga una disminución del tamaño de los BDD. En los circuitos grandes, unas pocas particiones disjuntas situadas en lugares estratégicos pueden hacer disminuir extraordinariamente el tamaño de los BDD resultantes, además de no provocar un error significativo.

En consecuencia, se debe evitar realizar **particiones disjuntas exhaustivas**. Por partición disjunta exhaustiva se entiende a la partición del circuito que resulta de aplicar la partición disjunta en todos los puntos del circuito que sea posible. El resultado es una partición con el mayor número de regiones disjuntas posibles y, por lo tanto, las regiones tendrán un tamaño mínimo. En el capítulo de resultados experimentales se verá que esta estrategia de particionado no es conveniente por el aumento del error, y porque tener regiones tan pequeñas no ofrece ninguna ventaja práctica.

El otro tipo de estrategia para la partición son las **particiones disjuntas mínimas**. Por partición disjunta mínima se entiende a la partición del circuito que resulta de aplicar la partición disjunta exclusivamente en aquellos puntos donde:

- La partición ofrezca una disminución significativa del tamaño de los BDD.
- Las áreas resultantes de la partición mantengan un tamaño grande aunque manejable.
- El análisis de las condiciones de error que se han visto en este apartado sea suficientemente favorable.

En algunos circuitos de ejemplo del capítulo de resultados (capítulo 4) se analiza los errores y los tamaños de los BDD para sus particiones disjuntas exhaustiva y mínima. En el apartado AIII.3.1 del anexo se analizan estas particiones para un circuito concreto.

3.4.4 Conclusiones sobre la partición de los circuitos

Además de la partición en regiones reconvergentes, en esta tesis se propone la partición de los circuitos en regiones disjuntas. Las regiones disjuntas producidas por las sentencias condicionales de los lenguajes de descripción de hardware ofrecen un medio eficaz y exacto para el cálculo de la **probabilidad** de señal. Esto se debe a que en caso de dependencias estructurales, **la región disjunta está acotada a las señales de entrada del multiplexor** que se forma, mientras que la región de reconvergencia abarca todas las señales hasta las señales comunes (véanse los ejemplos del apartado 3.4.1.2). Estas regiones de reconvergencia pueden llegar hasta las entradas del circuito, lo que puede provocar un aumento excesivo en las necesidades de cálculo. Y por tanto, como se verá en el capítulo 4, **la reducción de área lograda por la partición en regiones disjuntas puede ser muy significativa**.

Sin embargo, en el apartado 3.4.2 (y el anexo AII.2) se ha mostrado que el cálculo de la **actividad** con regiones disjuntas es aproximado cuando existen reconvergencias que afectan a las entradas del multiplexor. Aún así, el error cometido en el cálculo de la actividad se puede asumir cuando se cumple algunas de las condiciones expuestas en el apartado 3.4.3 y se sigue una estrategia con **particiones disjuntas mínimas**.

3.5 Construcción de los BDD

En los anteriores apartados se ha explicado la elaboración del modelo del circuito (EHM §3.3) y posteriormente la partición de dicho modelo (§3.4). Por lo tanto, se dispone de un modelo sobre el que trabajar, que se ha dividido para reducir los requerimientos computacionales y de memoria.

Las ecuaciones de probabilidad y actividad de los nodos del EHM se obtienen mediante diagramas de decisión binaria (BDD). Se puede recurrir a otros métodos distintos a los BDD, como la representación polinómica [103] y las redes bayesianas [6]. Sin embargo se ha optado por los BDD por su amplio uso y por la existencia de herramientas de código abierto para la manipulación de BDD.

Previamente a la obtención de la ecuación de actividad de cada nodo, se deben obtener el valor de probabilidad y actividad de los nodos de los que depende en su partición.

Los fundamentos de la obtención de las **ecuaciones de probabilidad** a partir del BDD de probabilidad se explicaron en el apartado 2.2.2. Por otro lado, en el anexo AII.3 se explica el mecanismo de **construcción de los BDD de probabilidad** a partir del EHM. Esta explicación se ha puesto en el anexo para simplificar el desarrollo de esta sección, ya que si bien puede resultar interesante, no es necesaria para su comprensión.

El objetivo de esta sección es exponer la obtención de las **ecuaciones de actividad** y la **construcción de los BDD de actividad**.

El **cálculo de la actividad** es bastante más **complejo** que el de la probabilidad debido a la consideración de las **dependencias temporales** de las señales (§2.2.3.2), y por ello, tanto el tamaño de las ecuaciones como el de los BDD se incrementan de manera significativa.

A raíz de esto, **en esta tesis se propone una nueva manera de representar la actividad en un BDD**. Con esta propuesta se reduce entre un 25% y un 50% el tamaño de los BDD. Antes de explicar esta nueva propuesta, en el siguiente subapartado se profundizará en la solución más comúnmente empleada. Posteriormente, en el apartado 3.5.2 se expondrá la solución propuesta. Por último, en los apartados 3.5.3 y 3.5.4 se comparan ambas alternativas.

3.5.1 BDD de actividad basado en tiempo (TFBDD)

En esta sección se explican los BDD de actividad basados en tiempo (TFBDD). Estos se utilizan para el cálculo de la actividad, y resultan de aplicar la operación XOR en dos tiempos consecutivos[35] (0, *T*) de la probabilidad de la señal (ecuación 2.28). Esta operación incrementa extraordinariamente el tamaño de la ecuación y los BDD de actividad. Los TFBDD fueron propuestos por *Schneider* [119] (§2.3, figura 2-25, pág. 41), y se explican en esta sección para compararlos con los BDD de actividad que se proponen en esta tesis (§3.5.2).

A continuación se explicará la estructura e interpretación de los TFBDD (§3.5.1.1) y en el siguiente apartado (§3.5.1.2) se expondrá cómo se obtiene la ecuación de actividad a partir de los TFBDD.

3.5.1.1 Estructura de los TFBDD

Como ya se introdujo en el apartado 2.3, el resultado de la aplicación de la XOR al BDD de probabilidad en dos tiempos consecutivos es un BDD basado en tiempos. El TFBDD resultante se ordena de manera que se mantiene el orden de las variables del BDD original pero a cada variable correspondiente a una señal en tiempo 0 le sigue la variable de la misma señal en tiempo *T*. Estas nuevas variables correspondientes al tiempo *T* tienen que ser creadas. En el

[35] Se supone que hay un ciclo mínimo de reloj *T*

TFBDD, los nodos correspondientes a la misma señal pero de tiempos diferentes se evalúan de forma conjunta.

Antes de continuar se va a hacer un paréntesis para especificar la nomenclatura y no confundir señales, nodos y variables. En la figura 3-27 se muestra una representación gráfica de cada uno de estos elementos (es el mismo TFBDD que el de la figura 2-25):

- **Señales** son las salidas y entradas de los elementos lógicos y estructurales de un circuito, muchas veces se les denomina nodos del circuito, sin embargo, en esta tesis se evitará llamarlos así para no confundirlos con los nodos de los BDD
- **Nodos** de un BDD son los círculos del BDD, también llamados vértices. En los BDD siempre hay dos nodos terminales o finales, que se representan con un cuadrado. A efectos prácticos de contabilizar el número de nodos, estos dos nodos terminales no se tendrán en cuenta.
- **Variables** de un BDD son las distintas etiquetas que figuran dentro de los nodos del BDD. Nótese que para BDD basados en tiempos puede haber más de una variable para una señal del circuito. Por ejemplo, en la figura 3-27, para la señal *A* hay dos variables en el TFBDD: A^0 y A^T.

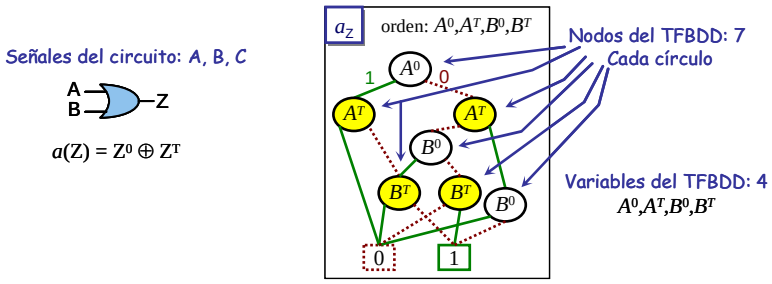


Figura 3-27: Diferencia entre señales de un circuito, nodos y variables de un BDD

La estructura general de un TFBDD para una señal cualquiera *A* es la mostrada en figura 3-28. En ella está recuadrada la parte concerniente a la señal *A*, encima del nodo A^0 hay un nodo cualquiera x^t de otra señal en un tiempo cualquiera (0 ó *T*); y de la misma manera, bajo los nodos A^T hay cualquier otro nodo, que puede ser diferente para cada una de las ramas. Lo importante ahora es el significado del recuadro, en él, tras recorrer hacia abajo los nodos desde A^0, se obtienen las cuatro posibilidades de transición de la señal *A*: $A_{0\to0}$, $A_{1\to1}$, $A_{0\to1}$, $A_{1\to0}$.

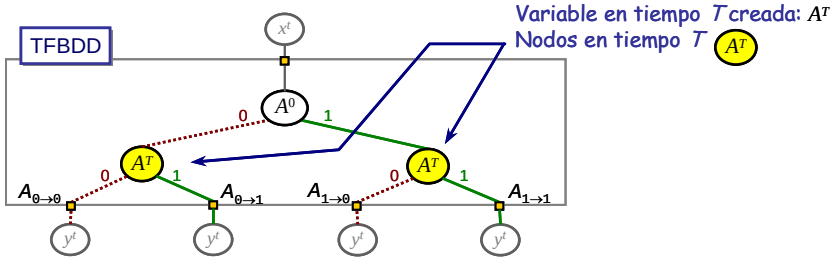


Figura 3-28: Estructura general de un TFBDD para la señal A

Estas transiciones se obtienen evaluando conjuntamente cada uno los cuatro caminos posibles que partiendo desde A^0 llegan a las cuatro terminaciones del recuadro (figura 3-29). Por ejemplo, la rama cero de A^0 implica que $A(0) = 0$ y la rama uno de A^T significa $A(T) = 1$. Por tanto, recorriendo el diagrama desde A^0 por estas dos ramas se obtiene $A_{0\to1} = \{(A^0 = 0) \wedge (A^T = 1)\}$.

En la derecha de la figura 3-29 se muestran separados cada uno de los cuatro caminos que se pueden recorrer.

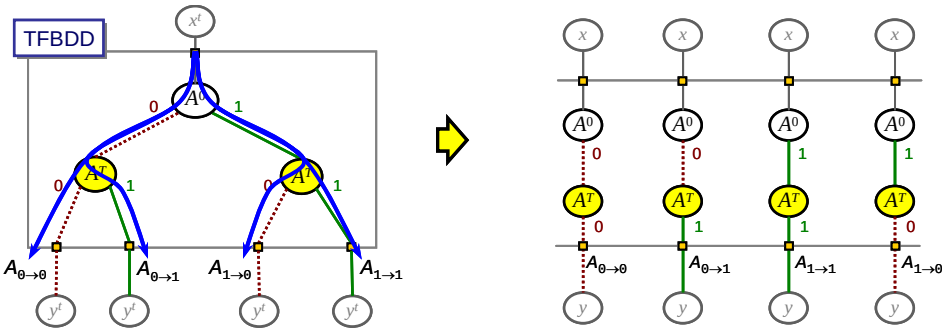


Figura 3-29: Los cuatro caminos de la estructura general de un TFBDD para la señal A

La figura 3-28 muestra la estructura general de los TFBDD para una señal, sin embargo, en ocasiones pueden aparecer estructuras simplificadas que surgen de las reglas de reducción de los BDD[36]. Estas reducciones disminuyen el número de nodos y caminos del BDD, y por tanto, su tamaño. Por otro lado, estas reducciones hacen surgir nuevas interpretaciones al recorrer los caminos.

En la figura 3-30 se muestra un ejemplo del proceso de reducción. En la estructura de la izquierda, las dos ramas de uno de los nodos A^T terminan en el mismo nodo. Esto hace que se pueda eliminar dicho nodo, y conectar directamente el nodo de arriba con el de abajo. Como resultado se tiene la estructura de la derecha de la figura 3-30.

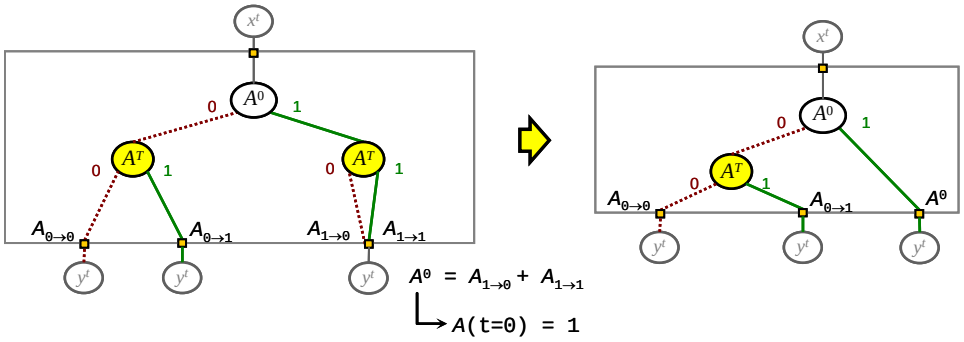


Figura 3-30: Reducción de un TFBDD y la interpretación del nuevo camino

Como consecuencia de la reducción surgen caminos diferentes a los mostrados en la figura 3-29. El significado de estos caminos se puede deducir analizando el proceso de reducción. Por ejemplo, en la figura 3-30 tenemos que los caminos $A_{1\to0}$ y $A_{1\to1}$ se encuentran. La unión de estos dos eventos es[37]:

$$\{[(A^0{=}1) \wedge (A^T{=}0)\,] \vee [(A^0{=}1) \wedge (A^T{=}1)]\} =$$
$$= \{[(A^0{=}1) \wedge [(A^T{=}0) \vee (A^T{=}1)]\} = \{A^0{=}1\}$$

Que puesto de otra manera más clara[38]:

$$A^0 \cdot \overline{A^T} + A^0 \cdot A^T = A^0 \cdot (\overline{A^T} + A^T) = A^0$$

Que es la interpretación de recorrer ese camino.

En la figura 3-31 se muestran algunos ejemplos de estructuras simplificadas.

[36] Recuérdese de la sección 2.2.2 que lo que en esta tesis se denomina BDD, estrictamente son ROBDD, esto es, BDD reducidos y ordenados.

[37] Recuérdese que:
$A^0 \equiv A(t{=}0)$; $A^T \equiv A(t{=}T)$;

[38] Recuérdese que al referirse a eventos:
$A^0 \equiv \{A(t{=}0) = 1\}$; $A^T \equiv \{A(t{=}T) = 1\}$; $\overline{A^T} \equiv \{A(t{=}T) = 0\}$

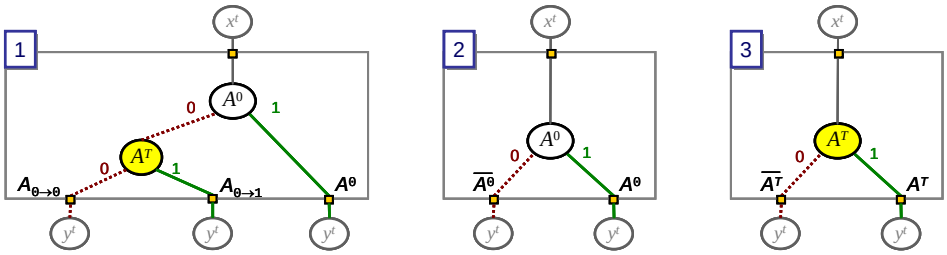


Figura 3-31: Varias estructuras simplificadas de TFBDD para la señal A

La interpretación de todos los posibles caminos que pueden surgir de las estructuras TFBDD de una señal se han representado en la figura 3-32. Por ejemplo, la estructura 1 de la figura 3-31 tiene los caminos primero, segundo y quinto de la figura 3-32 (empezando a contar por la izquierda). La estructura 3 de la figura 3-31 tiene los dos últimos caminos de la derecha de la figura 3-32.

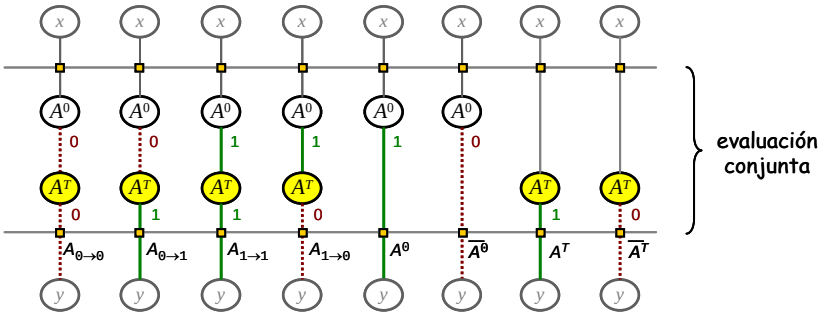


Figura 3-32: Caminos que pueden aparecer en la estructura de un TFBDD para una señal

Una vez que se ha expuesto la estructura e interpretación de los TFBDD ya se puede explicar cómo se obtiene la ecuación de la actividad a partir de un TFBDD. En la siguiente sección se detalla este proceso. También se explica brevemente cómo se crea el TFBDD.

3.5.1.2 Obtención de la ecuación de actividad

La obtención de la ecuación de actividad a partir de un TFBDD se realiza teniendo presente que, en cada camino, los nodos de una misma señal se deben evaluar conjuntamente. Como las señales de entrada se consideran independientes y el análisis de reconvergencia (y en su caso de regiones disjuntas) se ha realizado, se puede admitir que las señales de un TFBDD son independientes entre sí, y por tanto sus probabilidades se pueden separar.

Para explicarlo se utilizará un ejemplo, que complementará al visto en la figura 2-25. En la figura 3-33 se ilustra el proceso de obtención del TFBDD para la señal de salida (Z) de un multiplexor. Para ello, se aplica la operación XOR entre los BDD de la salida en dos tiempos consecutivos: $Z^0 \oplus Z^T$, lo que proporciona la ecuación de la actividad de la salida: $a(Z)$.

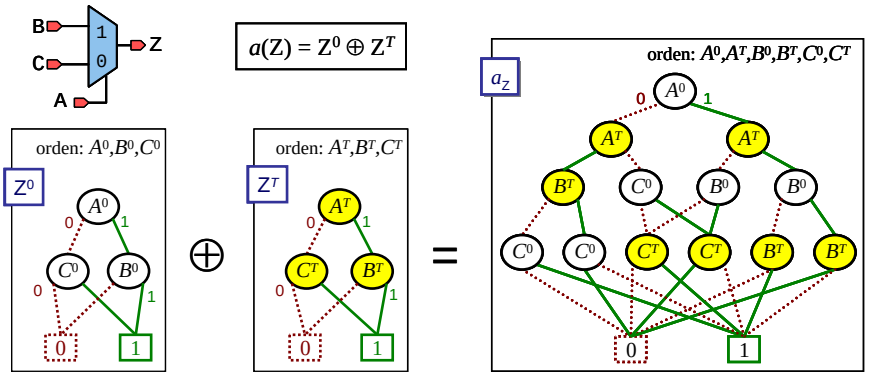


Figura 3-33: Obtención del TFBDD de un multiplexor aplicando la XOR a los BDD de la salida Z *en dos tiempos consecutivos* (0, *T*)

Obsérvese que el orden del TFBDD conserva el orden del BDD original, aunque manteniendo juntas las variables de la misma señal en distinto tiempo. Así, si el orden original era {*A*, *B*, *C*} el orden del TFBDD es {A^0, A^T, B^0, B^T, C^0, C^T}.

Los pasos requeridos para la creación del TFBDD son:

- Para cada variable del BDD se crea una variable de la misma señal en tiempo *T*.
- Establecimiento del nuevo orden de las variables del BDD: El orden entre variables de distinta señal permanece igual. Pero para las variables de un misma señal, la variable en tiempo *T* va a continuación de la variable en tiempo 0
- Creación de una copia del BDD pero con las variables relativas al tiempo *T* (el BDD original se considera en tiempo 0).
- Aplicar la operación XOR entre los BDD en tiempo 0 y tiempo *T*

Una vez que se tiene el TFBDD, la ecuación de actividad se obtiene recorriendo todos los caminos que llevan al nodo final uno. En la figura 3-34 aparecen cada uno de estos caminos para el ejemplo de la figura 3-33. Debajo de cada uno de los caminos se ha puesto su significado. Nótese cómo en los caminos donde aparecen los dos nodos de una misma señal en sus dos tiempos (0, *T*), la evaluación es conjunta.

Por las propiedades de los BDD, cada camino es disjunto del resto, y por tanto se puede calcular su probabilidad separadamente. Además, como las señales se consideran independientes, dentro de cada camino se puede separar la probabilidad de cada señal, y así se obtiene la ecuación que aparece en la parte baja de la figura 3-34.

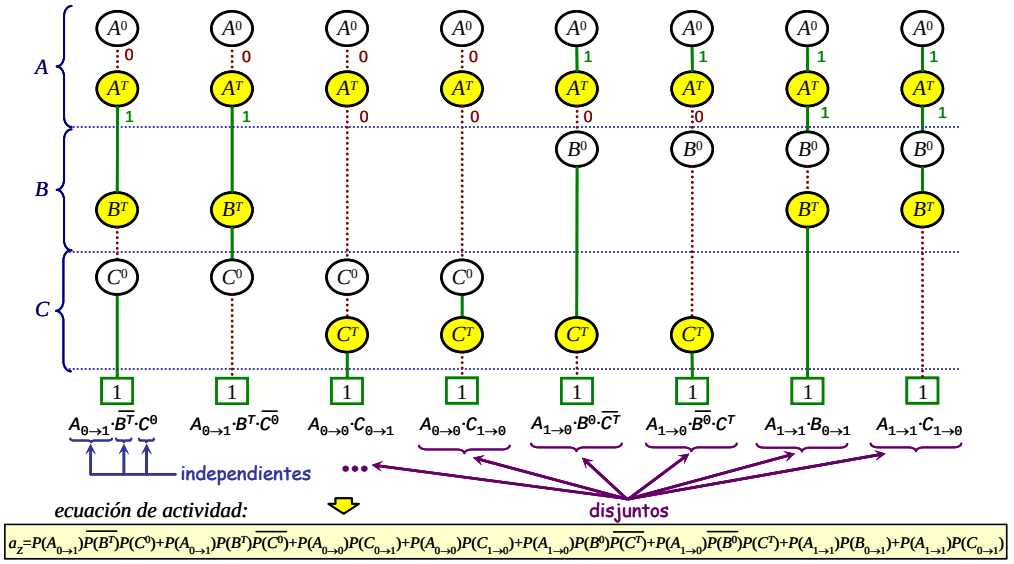


Figura 3-34: Obtención de la ecuación de actividad del multiplexor a partir de los caminos que terminan en el nodo uno de su TFBDD (ver figura 3-33)

Aunque para realizar los cálculos no es necesario extraer la ecuación de actividad, la obtención de ésta se ha explicado para ayudar a comprender el significado de los TFBDD y facilitar la comparación con los BDD de actividad que se proponen en esta tesis. El algoritmo de cálculo de la probabilidad a partir un BDD se explica en el anexo AI.2. Este algoritmo se adapta al TFBDD de modo que los nodos de una misma variable se evalúen de manera conjunta.

En el siguiente apartado se explicará el BDD de actividad propuesto en esta tesis. La información proporcionada en este apartado permitirá compararlo con los TFBDD.

3.5.2 BDD de actividad propuesto

Con el objeto de **disminuir el tamaño de los BDD de actividad**, en esta tesis doctoral **se propone una nueva forma de crearlos y representarlos** con la que se logran **importantes reducciones en los tamaños** de los BDD.

En la propuesta, **en vez de emplear una representación basada en tiempos**: x^0 y x^T, como se hace en los TFBDD; **se propone una representación basada primero en actividad e inactividad**: a_x y $\overline{a}_x$; y luego, **en probabilidades de transición**: $P(x_{0\to0})$, $P(x_{1\to1})$, $P(x_{0\to1})$, $P(x_{1\to0})$. Este tipo de representación se ha denominado *BDD de actividad*, o en su forma abreviada: ***a*BDD**.

A continuación se expondrá la estructura del *a*BDD propuesto (§3.5.2.1), **esta estructura es una propuesta original de la tesis**. Luego se compararán las estructuras de los *a*BDD con los TFBDD (§3.5.2.2). En la sección 3.5.2.4 se definirá una función de actividad para los BDD. Esta **definición de la función de actividad** es también **un desarrollo original de la tesis**, y es útil para crear los *a*BDD a partir de los BDD de probabilidad. Los BDD tienen definidos diversos operadores, sin embargo, al ser los *a*BDD una estructura nueva, es importante disponer de un operador que, de manera eficiente, los cree a partir de los BDD de probabilidad. Posteriormente, en el apartado 3.5.2.5 se detalla la obtención de la actividad a partir de los *a*BDD.

Ya fuera de esta sección, en los apartados 3.5.3 y 3.5.4 se comparan los BDD de actividad para distintas funciones lógicas y distinto número de entradas.

3.5.2.1 Estructura de los *a*BDD

La estructura general de un *a*BDD para la variable *A* se muestra en la figura 3-35, en ella, en vez de crear una variable de la señal en un tiempo consecutivo (A^T); se crea una variable para la

actividad a_A ([39]). Respecto al orden de las variables del *a*BDD, se mantiene el mismo orden de variables de distinta señal, pero se pone la variable de la actividad de una señal inmediatamente antes que la variable de su misma señal.

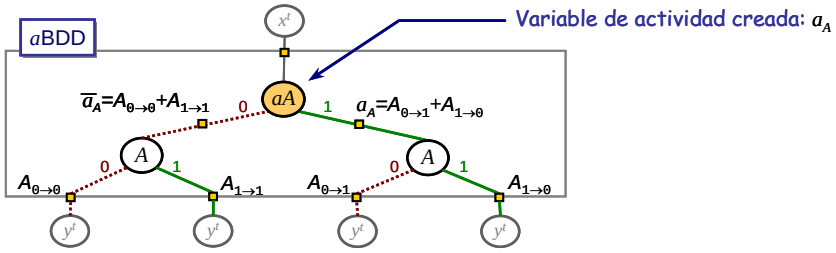


Figura 3-35: Estructura general de un aBDD para la señal A

De la figura 3-35 se observa que de la rama cero del nodo de actividad se obtiene la inactividad de la señal, $\overline{a}_A$. Por otro lado, de la rama uno del nodo de actividad se llega a la actividad de la señal a_A. Recuérdese de la ecuación 2.6 que la fórmula de la inactividad de la señal es:

$$\overline{a}_A = P(A_{0\to0}) + P(A_{1\to1})$$

Y que por la ecuación 2.5, la actividad de la señal es:

$$a_A = P(A_{0\to1}) + P(A_{1\to0})$$

Lo que hace el nodo de la variable *A* cuando sigue al nodo de actividad es separar la actividad e inactividad en los sumandos que aparecen en estas fórmulas. Así, por ejemplo, recorrer el camino por la rama cero del nodo de actividad y luego tomar la rama cero del nodo de la variable, implica que la señal se mantiene en valor cero durante dos ciclos consecutivos, esto es: $A_{0\to0}$. Si en vez de tomar la rama cero del nodo de la variable, se hubiese tomado la rama uno, sería $A_{1\to1}$, que también forma parte de la inactividad.

Tomando la rama uno del nodo de actividad se llega a las dos posibilidades de transición de la señal con cambio de valor: $A_{0\to1}$, $A_{1\to0}$; que son las componentes de la actividad de la señal.

De la misma manera que en los TFBDD no siempre aparece la estructura general mostrada en la figura 3-35, sino que pueden surgir estructuras más simples como las mostradas en la figura 3-36.

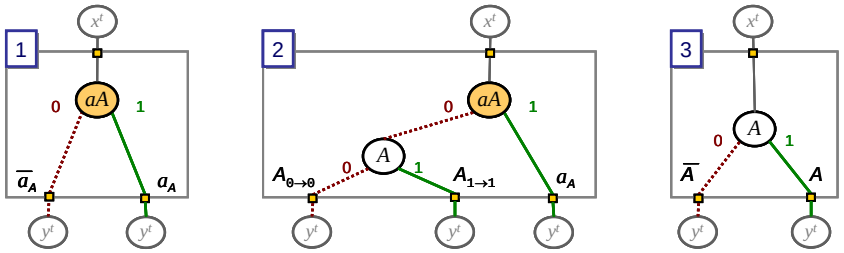


Figura 3-36: Varias estructuras simplificadas de aBDD para la señal A

Todos los posibles caminos que pueden aparecer de las estructuras *a*BDD de una señal se han representado en la figura 3-37. En el dibujo, al terminar el camino se ha indicado su interpretación. Por ejemplo, la primera estructura de la figura 3-36 tiene los caminos quinto y sexto de la figura 3-37. La segunda estructura de la figura 3-36 tiene los caminos primero, segundo y quinto de la figura 3-37. Y la tercera estructura de la figura 3-36 tiene los dos últimos caminos de la figura 3-37.

[39] En las figuras, las variables aparecen como *aA*, en vez de a_A, para evitar letras muy pequeñas y así facilitar la lectura.

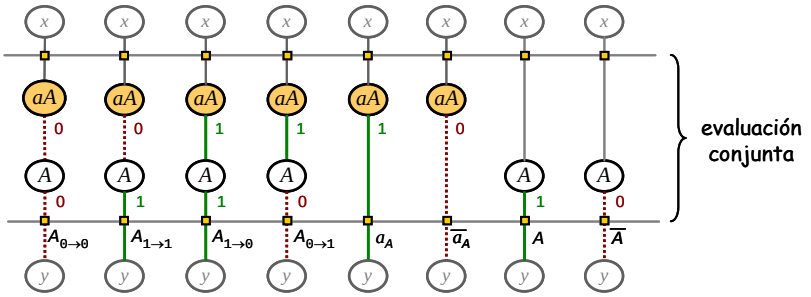


Figura 3-37: Caminos que pueden aparecer en la estructura aBDD de una señal

Considerando la ecuación 2.4, por la que las probabilidades de transición con cambio de valor son iguales: $P(A_{0\to1}) = P(A_{1\to0}) = \frac{1}{2}a_A$, se pueden obtener representaciones de *a*BDD más compactas. En esta representación, en vez de tomar la estructura general de la figura 3-35, se escoge una en la que no se hace diferenciación entre $A_{0\to1}$ y $A_{1\to0}$. Por tanto, se sustituyen $P(A_{0\to1})$ y $P(A_{1\to0})$ por $\frac{1}{2}a_A$. Así, la estructura general para una señal *A* queda como la mostrada en la figura 3-38, en la que la que las ramas cero y uno del nodo de *A* que sale de la rama uno del nodo de actividad están indiferenciadas, y por tanto en vez de llegar a $P(A_{0\to1})$ ó a $P(A_{1\to0})$, por ambos caminos se llega a $\frac{1}{2}a_A$.

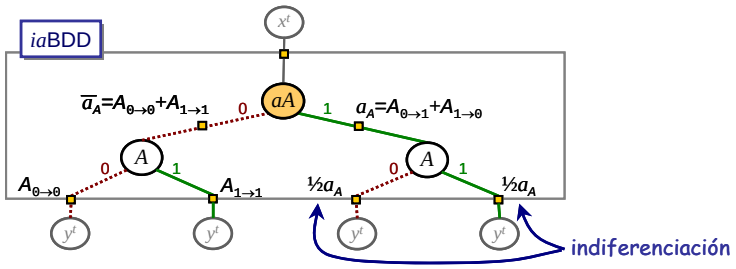


Figura 3-38: Estructura general de un aBDD indeferenciado para la señal A

A esta alternativa de representación se llamará *aBDD indiferenciado* (*ia*BDD), en contraposición con el anterior que se denominará *a*BDD *diferenciado* (*da*BDD). De todos modos se seguirán llamando *a*BDD cuando no sea necesario la distinción, y en cualquier caso, **si no se especifica se entenderá que se trata del *a*BDD indiferenciado** (*ia*BDD), que son los que se emplearán en esta tesis.

Manteniendo las condiciones especificadas en el apartado 3.2, el uso del *ia*BDD en vez del *da*BDD produce los mismos resultados numéricos, ya que el valor de la evaluación conjunta para cada señal es el mismo. La figura 3-39 muestra la interpretación de los caminos posibles para un *ia*BDD. Obsérvese que son todos iguales menos dos que están marcados con una flecha. Estos dos caminos no están diferenciados, pero su valor numérico es el mismo que para sus equivalentes del *da*BDD (compárese con la figura 3-37).

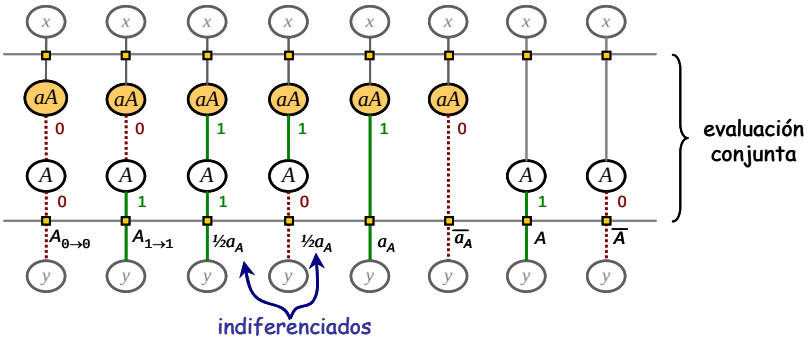


Figura 3-39: Caminos que pueden aparecer en la estructura iaBDD de una señal

El empleo de caminos no diferenciados puede dar lugar a importantes reducciones en el número de nodos totales de un BDD. Esto se logra debido a que proporciona un mayor grado de libertad que es aprovechado para simplificar los diagramas.

El empleo de los *da*BDD puede ser interesante cuando se desea conservar la información del tipo de transición: 0→1 ó 1→0; aunque la probabilidad de estas transiciones sea la misma. En los casos en que se necesite considerar las dependencias espaciales entre las señales, el conocimiento del tipo de transición (subida o bajada) es importante porque estas transiciones pueden condicionar de manera diferente al resto de señales. Por ejemplo, en un caso con señales fuertemente correlacionadas como son los bits de un contador, la transición de subida (0→1) del bit cero no origina actividad en el bit uno, sin embargo, la transición de bajada (1→0) provoca actividad en el bit uno. Estas transiciones no se distinguirían con los *ia*BDD pero sí con los *da*BDD.

Tanto **los *a*BDD indiferenciados como los diferenciados son una propuesta original de esta tesis**. Sin embargo, en esta tesis se emplearán los indiferenciados salvo que se especifique lo contrario. En el anexo AII.4 se analizan con más detalle las diferencias entre los *ia*BDD y los *da*BDD, además, en la sección 3.5.4.3 se muestran ejemplos de diferencias de tamaños entre los BDD de estas dos alternativas.

3.5.2.2 Comparación de las estructuras de los TFBDD y *aBDD*

Con los *a*BDD se consigue una notable reducción del número de nodos del BDD de actividad, esto se debe a las diferencias entre las estructuras básicas de cada uno de ellos.

Las estructuras generales de los TFBDD y los *a*BDD no difieren en cuanto al número de nodos (compárese las figuras 3-28 y 3-35). Sin embargo, en muchas estructuras simplificadas la representación mediante *a*BDD resulta más ventajosa.

La figura 3-40 muestra un caso muy habitual en los BDD de actividad. En éste, no hay diferencia entre el camino que obtiene la transición 0→1 con el de la transición 1→0, y por tanto, en el TFBDD ambos caminos coinciden en el mismo nodo. La unión de ambas transiciones representa la actividad de conmutación, y eso es precisamente lo que se aprovecha en los *a*BDD, en los que de la rama uno del nodo de actividad se obtiene directamente la actividad. Con esto se consigue el ahorro de un nodo del total de tres que tenía el TFBDD, y de un camino de cuatro.

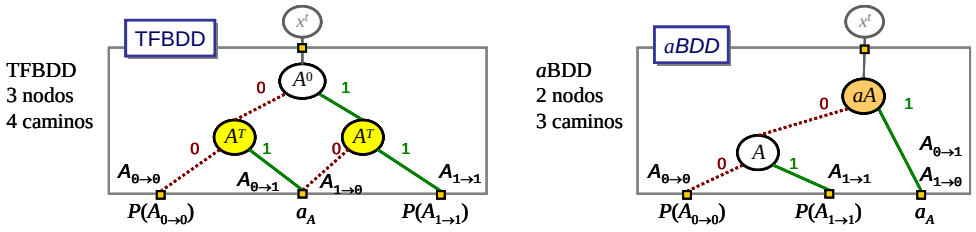


Figura 3-40: Comparación entre TFBDD y aBDD cuando las transiciones 0→1 y 1→0 terminan en el mismo nodo

Una situación todavía más ventajosa es aquella en la que también las transiciones 0→0 y 1→1 terminan en el mismo nodo, aunque diferente al de las transiciones 0→1 y 0→1. En este caso, la unión de ambas representa la inactividad de la señal, lo que es aprovechado por los *a*BDD, que la obtienen directamente de la rama cero del nodo de actividad. La figura 3-41 muestra la diferencia entre ambas estructuras, donde se reducen dos nodos de tres, y dos caminos de cuatro.

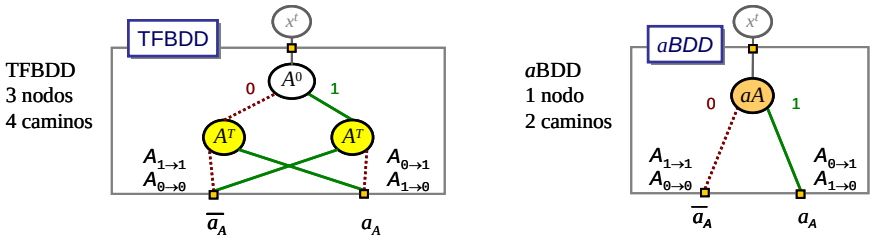


Figura 3-41: Comparación entre TFBDD y aBDD cuando las transiciones se reducen a actividad e inactividad

Como último ejemplo, la figura 3-42 muestra cómo se representa la probabilidad de señal en los TFBDD y *a*BDD. En los TFBDD existen dos representaciones posibles, una para la probabilidad en tiempo 0 y otra para la del tiempo *T*. Sin embargo, estas probabilidades son iguales y ambas se pueden sustituir por una representación sin tiempos, como la de los *a*BDD. Con esto se consigue un ahorro de un nodo (en estructuras diferentes) respecto de dos nodos posibles, y un ahorro de dos caminos frente a cuatro.

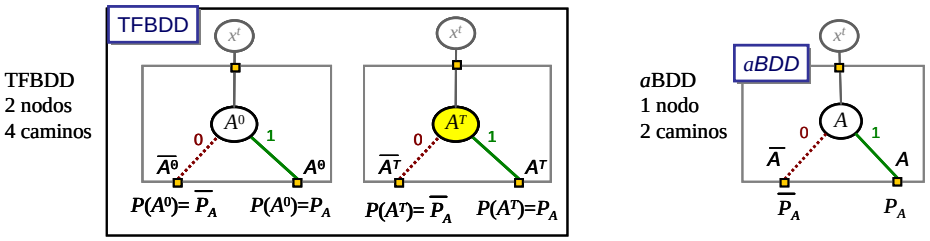


Figura 3-42: Comparación entre los TFBDD y aBDD que indican la probabilidad de señal

Hay más situaciones en las que se logran reducciones de tamaño, pero no se van a describir debido a que se tratan de combinaciones de los casos que se acaban de describir.

Es importante señalar que **todas las estructuras de aBDD tienen un número igual o menor que la estructura correspondiente en TFBDD**, por tanto, nunca será mayor el *a*BDD resultante que el TFBDD. En la práctica, de la multitud de pruebas realizadas (ver secciones 3.5.3, 3.5.4 y 4.4.1), **siempre se ha conseguido una considerable reducción del tamaño del BDD**.

3.5.2.3 Transformación de TFBDD en *a*BDD

Todo TFBDD se puede convertir en un *a*BDD mediante un conjunto de transformaciones elementales basadas en las comparaciones del apartado anterior. Sin embargo, la transformación inversa, de *a*BDD a TFBDD, no es única debido a que en los *a*BDD se realizan simplificaciones que ocasionan pérdida de información. Bajo las condiciones del modelo propuesto (§3.2), esta pérdida de información no repercute en el cálculo de la actividad.

Por ejemplo, la reducción mostrada en la figura 3-42 implica una pérdida de información irreversible, de modo que a partir de la estructura del *a*BDD de la figura no se puede saber de cuál de los dos TFBDD provenía.

La explicación de la transformación de los TFBDD en *a*BDD se realizará a través de un ejemplo. Se partirá del TFBDD de una puerta OR, cuya formación se ha mostrado en la figura 2-25.

A la izquierda de la figura 3-43 se muestra el TFBDD original de la puerta OR. En ese TFBDD se ha señalado con una flecha un nodo temporal B^0. Como se puede observar en la figura central, este nodo no tiene más nodos de la misma señal *B* arriba ni abajo, y es por eso por lo que se ha recuadrado, formando una estructura básica de TFBDD. Por tanto, ese recuadro se puede sustituir por la estructura equivalente en *a*BDD (igual que en la figura 3-42).

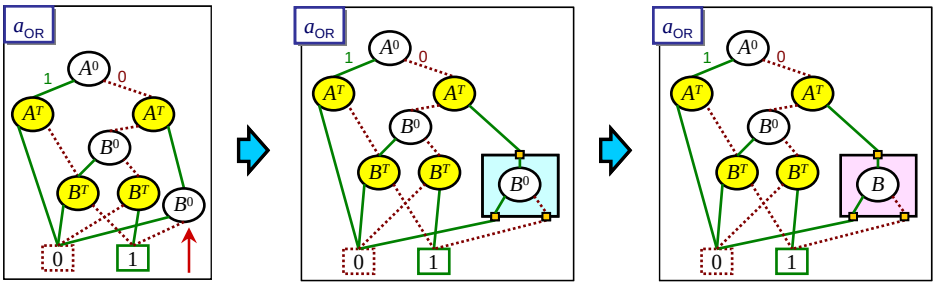


Figura 3-43: Paso 1, transformación de nodo temporal en nodo atemporal

Partiendo del resultado de la figura 3-43, el siguiente paso será duplicar el nodo B^T señalado con una flecha en la figura 3-44. Como a este nodo se accede desde dos ramas, este paso se realiza para analizar independientemente los caminos que llegan al nodo. El resultado de este paso es el BDD de la derecha de la figura 3-44, para facilitar su identificación, ambos nodos se han insertado en un rectángulo redondeado.

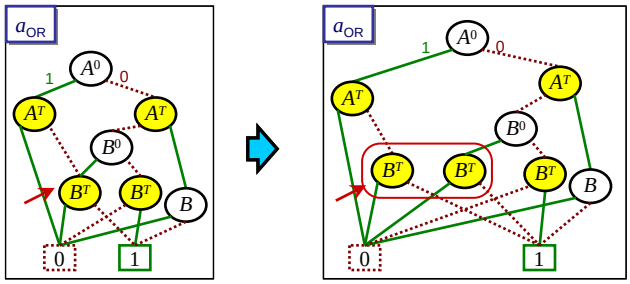


Figura 3-44: Paso 2, separación del nodo B^T en dos

Uno de los nodos resultantes del paso anterior se puede transformar en un nodo atemporal. Este nodo está señalado con una flecha en el BDD de la izquierda de la figura 3-45. Y en el BDD del medio se ha recuadrado dicho nodo para señalar que se puede sustituir según la transformación de la figura 3-42.

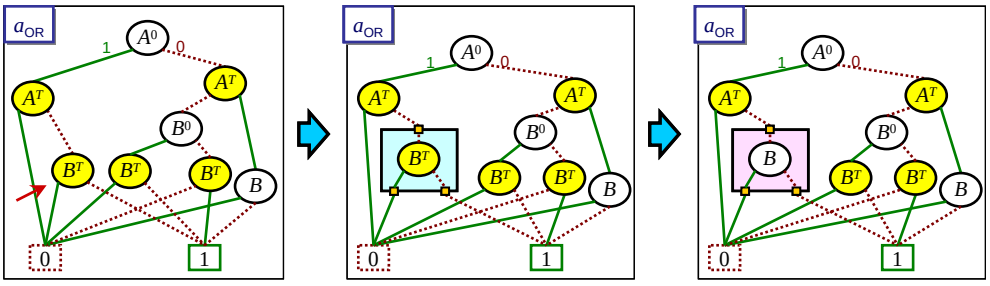


Figura 3-45: Paso 3, transformación de nodo temporal en atemporal

El nodo resultante de esta última operación y el resultante de la figura 3-43 son equivalentes[40], y por tanto se pueden unir en un único nodo. En el BDD de la izquierda de la figura 3-46 se han señalado ambos nodos con flechas; en el de la derecha está el resultado de la agrupación (señalado también con una flecha).

[40] Para que dos nodos de un BDD se consideren equivalentes, no sólo tienen que ser nodos de la misma variable, sino que todo el diagrama que a partir de ellos se forma hasta llegar a los nodos finales (uno y cero) ha de ser igual.

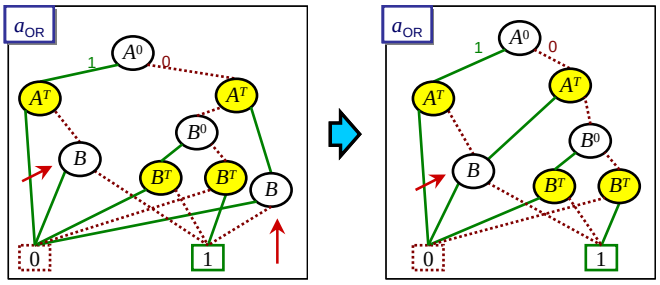


Figura 3-46: Paso 4, reducción de dos nodos equivalentes en uno

En el BDD de la izquierda de la figura 3-47 se han marcado un conjunto de nodos que, como se muestra en el BDD central, su estructura es equivalente al de la figura 3-41. Por consiguiente, esta estructura se puede sustituir por el nodo de actividad, dando como resultado el BDD de la derecha de la figura 3-47.

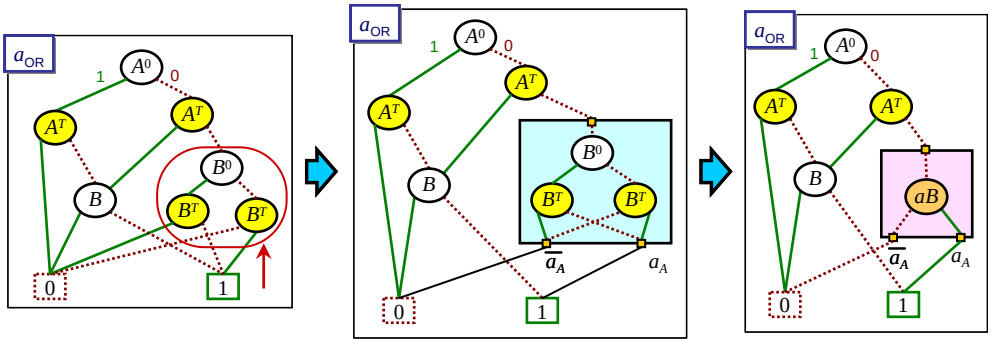


Figura 3-47: Paso 5, sustitución por nodo de actividad

La última transformación corresponde a los nodos de la señal *A*, que se han señalado en el BDD de la izquierda de la figura 3-48. En el BDD del medio se puede apreciar que se corresponde con una estructura equivalente a la de la figura 3-40 y por tanto se sustituye por la estructura *a*BDD equivalente. El resultado de esta sustitución se ha incluido a la derecha de la figura 3-48.

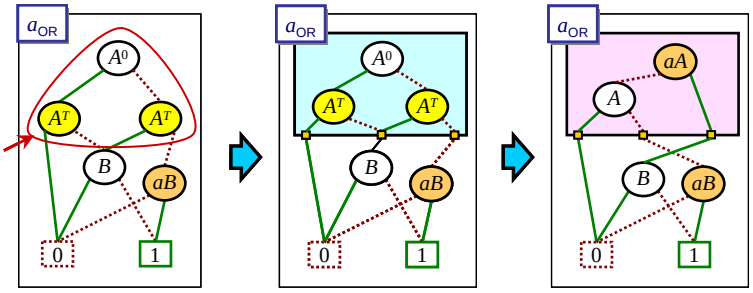


Figura 3-48: Paso 6, sustitución de estructura con transiciones 0→1 y 1→0 que terminan en el mismo nodo

Como resumen de la transformación y para comparar los BDD de actividad en las formas TFBDD y *a*BDD, éstas se han dibujado en la figura 3-49. Se puede apreciar que el *a*BDD es más sencillo, teniendo 4 nodos frente a los 7 nodos del TFBDD (reducción de un 43%). También hay una importante reducción del número de caminos, pasando de 9 caminos totales a 5 caminos (44% de reducción). En la figura también se ha mostrado el número de caminos que terminan en el nodo uno: 4 para el TFBDD frente a 2 para el *a*BDD; y el número de caminos que terminan en el nodo cero: 5 para el TFBDD frente a 3 para el *a*BDD.

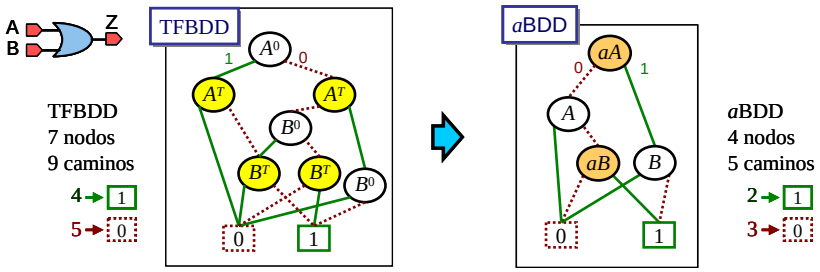


Figura 3-49: Comparación del TFBDD y aBDD para la puerta OR

Esta transformación de TFBDD a *a*BDD se ha implementado como función en la biblioteca BuDDy[64] y se ha integrado dentro de la herramienta Ardid [132], de manera que se pueda comparar automáticamente el tamaño de ambas representaciones de los BDD de actividad de los circuitos.

3.5.2.4 Definición de un operador actividad

En la sección anterior se ha explicado el procedimiento de la transformación de un TFBDD en un *a*BDD. Sin embargo, resulta poco útil idear una representación optimizada de los BDD de actividad si para llegar a ella es necesario pasar por la representación previa (los TFBDD). Por tanto, **en esta tesis se ha definido e implementado un operador actividad que crea los *a*BDD a partir de los BDD de probabilidad** sin recurrir a los TFBDD.

Hay diversas operaciones definidas para los BDD, como por ejemplo: AND, XOR, composición,... Como se ha creado un nuevo tipo de BDD, es necesario definir una nueva operación que lo cree, y con ello, especificar las propiedades de aplicar dicha operación. Posteriormente esta operación se podrá implementar como función en el programa informático que manipula los BDD.

El primer paso de la creación del *a*BDD es la definición del orden de las variables en el BDD. El nuevo orden es el mismo que el del BDD original, aunque las variables de la actividad de señal se intercalan entre cada variable de la señal. Así, si el orden de las variables del BDD es {*A*, *B*, *C*} el orden de las variables del *a*BDD será $\{a_A, A, a_B, B, a_C, C\}$.

Para la creación del *a*BDD sólo se necesita el BDD de probabilidad original. Es importante decir que el operador actividad se define para un sólo operando, aunque como es una función que se llama a sí misma al atravesar el BDD, las siguientes llamadas a la función pueden tener operandos distintos. Por tanto, aisladamente no tiene sentido aplicar el operador actividad a dos operandos diferentes, aunque sí lo tiene dentro de las sucesivas llamadas a la función.

Consecuentemente, el operador de actividad se define para dos operandos que son BDD de probabilidad ($\mathcal{M}$ y $\mathcal{N}$ son BDD de probabilidad, que se distinguirán por este tipo de letra cursiva) y el resultado es un *a*BDD (aQ, que se distinguirán también por la misma fuente cursiva, aunque precedidos por la a minúscula):

$$a[\mathcal{M},\mathcal{N}] \rightarrow aQ$$

Por tanto el dominio de la función de actividad son todos los BDD de probabilidad y su conjunto imagen son los *a*BDD. Este conjunto imagen contiene al dominio, pero no al contrario.

Cuando la función de actividad recibe un sólo operando ($\mathcal{M}$), se realiza la operación sobre ese mismo operando:

$$a[\mathcal{M}] = a[\mathcal{M},\mathcal{M}] \tag{3.7}$$

Para facilitar la compresión ya que no resulta fácil representar BDD en fórmulas, en muchas ocasiones se incluirán imágenes que representarán las operaciones. La ecuación 3.7 quedaría representada como la muestra la figura 3-50. Los operandos están dentro de un corchete y son BDD de probabilidad, para este caso, ambos operandos son el mismo debido a que se realiza el cálculo de la actividad de un BDD.

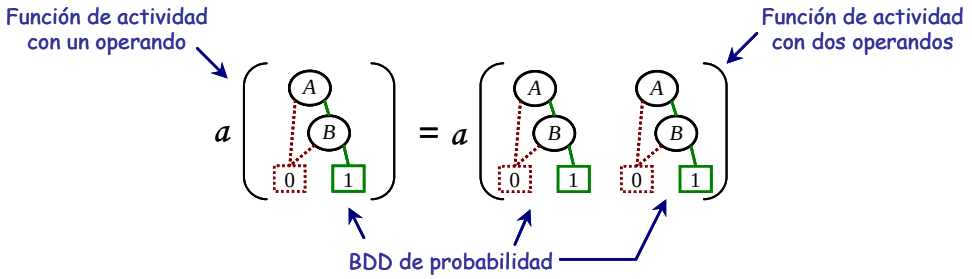


Figura 3-50: Equivalencia de la función de actividad para un operando

El operador de actividad es conmutativo:

$$a[\mathcal{M},\mathcal{N}] = a[\mathcal{M},\mathcal{N}]$$

No tiene sentido analizar las propiedades asociativa y distributiva para el operador de actividad, ya que los argumentos de la función de actividad (BDD de probabilidad) son diferentes al resultado, esto es, el conjunto imagen no pertenece al dominio de la función. Consecuentemente, el resultado de la función de actividad no puede volver a ser argumento de una función de actividad.

La función actividad de un BDD constante (*0* ó *1*) es el *a*BDD cero:

$$a[0] = a[1] = 0 \tag{3.8}$$

Por lo que también:

$$a[0,0] = a[1,1] = 0 \tag{3.9}$$

Representadas en imágenes, las ecuaciones 3.8 y 3.9 quedan como muestra la figura 3-51:

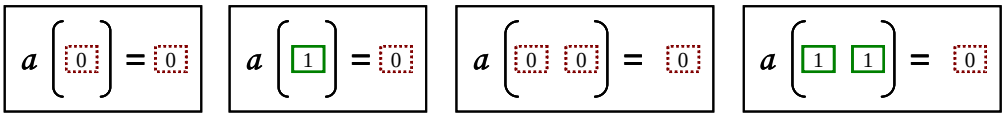


Figura 3-51: Función de actividad para BDD constantes e iguales

Por otro lado, la función de actividad de dos BDD constantes y diferentes es el *a*BDD unidad:

$$a[0,1] = a[1,0] = 1 \tag{3.10}$$

Que en imágenes la ecuación 3.10 se representa según la figura 3-52:

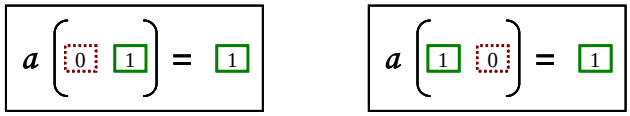


Figura 3-52: Función de actividad para BDD constantes y distintos

Los BDD unidad y cero pertenecen tanto al conjunto imagen (*a*BDD) como al dominio (BDD de probabilidad).

La operador de actividad aplicado a un nodo simple, esto es, un nodo que tanto su rama uno como su rama cero van a parar a nodos terminales y distintos (una rama al nodo cero y otra al uno), resulta en el nodo actividad de la señal del nodo simple.

$$a[\mathcal{A}] = a\mathcal{A} \tag{3.11}$$

Siendo $\mathcal{A}$ un BDD de probabilidad simple de la señal *A*, y $a\mathcal{A}$ el *a*BDD de la señal *A*. Para aclararlo, la ecuación 3.11 se muestra en la figura 3-53

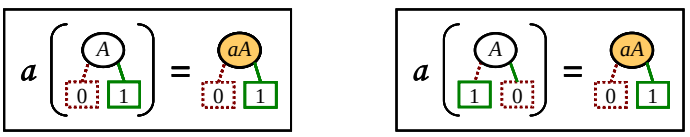


Figura 3-53: Función de actividad para BDD de probabilidad simples

La función de actividad para nodos simples de la variable *A*, pero teniendo uno negado y otro no, resulta la inactividad de *A*.

$$a[\mathcal{A}, \overline{\mathcal{A}}] = \overline{a\mathcal{A}} \tag{3.12}$$

Donde $\overline{\mathcal{A}}$ es un BDD de probabilidad simple que es el negado de $\mathcal{A}$, esto es, representa $\overline{P_A}$. Mientras que $\overline{a\mathcal{A}}$ es el *a*BDD de la inactividad de la señal *A*. Gráficamente la ecuación 3.12 queda ilustrada en la figura 3-54

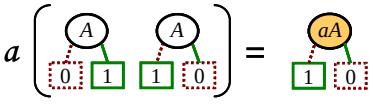


Figura 3-54: Función de actividad para BDD de probabilidad simples de una misma señal estando uno negado y otro no

La función de actividad de un BDD con el nodo terminal cero produce el mismo BDD:

$$a[\mathcal{M}, 0] = \mathcal{M} \tag{3.13}$$

La función de actividad con el nodo terminal uno produce el BDD negado.

$$a[\mathcal{M}, 1] = \overline{\mathcal{M}} \tag{3.14}$$

Estas propiedades abarcan a las ecuaciones 3.9 y 3.10, siendo por tanto el nodo terminal cero el elemento neutro del operador actividad. La operación con el nodo terminal uno no produce elemento simétrico (o inverso) porque la función de actividad de un nodo por su negado no da el elemento neutro (ecuación 3.12). Esto es, $\overline{\mathcal{M}}$ no es simétrico de $\mathcal{M}$.

Las ecuaciones 3.13 y 3.14 se representan en la figura 3-55

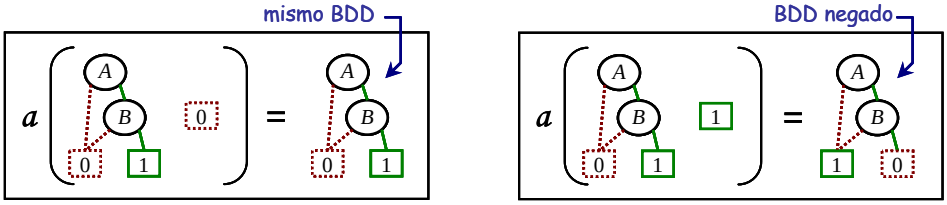


Figura 3-55: Función de actividad con BDD cero y uno

La función de actividad para dos BDD con **nodos raíz de distinta variable** se ejecuta recurrentemente a través de sus nodos hijos. Suponiendo que los operandos son los BDD $\mathcal{M}$ y $\mathcal{N}$, y que la variable del nodo raíz de $\mathcal{M}$ es anterior en orden a la variable del nodo raíz del $\mathcal{N}$, entonces:

- Se crea el nodo raíz del *a*BDD con la misma variable que la del nodo raíz de $\mathcal{M}$ (la variable con orden anterior)
- En la rama cero de nodo raíz se añade el resultado de la función de actividad entre $\mathcal{N}$ y el cofactor de $\mathcal{M}$ respecto a la variable negada del nodo raíz.
- En la rama uno se pone el resultado de la función de actividad entre $\mathcal{N}$ y el cofactor de $\mathcal{M}$ respecto a la variable del nodo raíz.

Puesto en fórmulas, asumiendo que la variable del nodo raíz de $\mathcal{M}$ es anterior a la de $\mathcal{N}$, la ecuación queda:

$$a[\mathcal{M}, \mathcal{N}] = \text{ITE}\left\{\text{raiz}(\mathcal{M}),\ a[\mathcal{M}_{\text{raiz}(\mathcal{M})}, \mathcal{N}],\ a[\mathcal{M}_{\overline{\text{raiz}(\mathcal{M})}}, \mathcal{N}]\right\} \tag{3.15}$$

Donde ITE(*A*, $\mathcal{R}$, $\mathcal{S}$) es la función *if-then-else* para los BDD. Y donde *A* es una variable BDD, mientras que $\mathcal{R}$ y $\mathcal{S}$ son BDD. Esta función construye un BDD creando el nodo raíz con la variable que hay en el primer argumento (*A*). En la rama uno del nodo raíz añade el BDD que hay en el segundo argumento ($\mathcal{R}$); mientras que en su rama cero pone el BDD que hay en el tercer argumento ($\mathcal{S}$).

Por otro lado, la función raiz($\mathcal{M}$) obtiene la variable del nodo raíz de $\mathcal{M}$. Por tanto, $\mathcal{M}_{raiz(\mathcal{M})}$ es el cofactor de $\mathcal{M}$ respecto a la variable de su nodo raíz, y $\mathcal{M}_{\overline{raiz(\mathcal{M})}}$ es el cofactor de $\mathcal{M}$ respecto a la variable de su nodo raíz negado.

La ecuación 3.15 puede ser difícil de comprender, para expresarlo de forma visual se ha incluido la figura 3-56. En ella se aplica la función de actividad a dos BDD $\mathcal{M}$ y $\mathcal{N}$ que tienen distinta variable en su nodo raíz. La variable del nodo raíz de $\mathcal{M}$ es *A*, que es anterior a *B*, que es la variable del nodo raíz de $\mathcal{N}$. Por tanto, se toma la variable *A* y se le aplica la función ITE. En cada una de las ramas se vuelve a aplicar la función de actividad, dejando en uno de los dos operandos a $\mathcal{N}$, mientras que en el otro operando se incluye el cofactor de $\mathcal{M}$ respecto a la variable del nodo raíz (*A*), que estará negado en la rama cero, y sin negar en la rama uno.

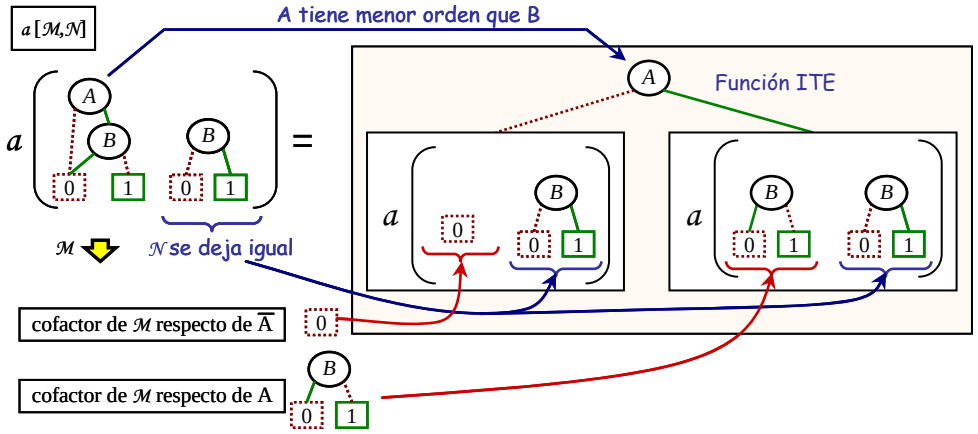


Figura 3-56: Función de actividad de BDD con nodos raíz de diferente variable

El siguiente paso se puede calcular a partir del mecanismo de la función de actividad que ya se ha descrito. Partiendo de la estructura resultante de la figura 3-56, se obtienen los BDD de cada una de las ramas. El de la rama cero por la ecuación 3.13 (figura 3-55), el de la rama uno por la ecuación 3.12 (figura 3-54). Este paso se ha detallado en la figure 3-57. En la derecha de la figura se muestra el *a*BDD resultante.

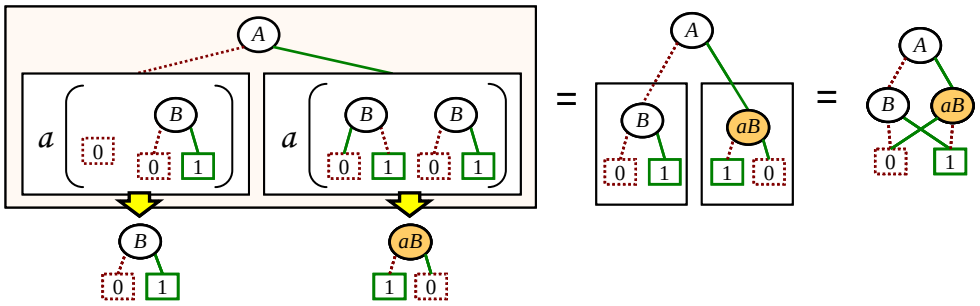


Figura 3-57: Continuación del cálculo del aBDD del ejemplo de la figura 3-56

Por último queda por explicar el cálculo de la función de actividad para dos BDD cuyas **variables del nodo raíz sean iguales**, éstos pueden tratarse del mismo BDD o no. La función de actividad crea la estructura general de un *a*BDD para la señal del nodo raíz (figura 3-38). De las cuatro ramas que salen de esta estructura general se van a colgar los *a*BDD resultantes de la aplicación de las funciones de actividad sobre combinaciones de cofactores de ambos BDD.

Se va a utilizar un ejemplo para ilustrar el proceso. En la figura 3-58 se muestra este ejemplo, donde el operando de la izquierda es $\mathcal{M}$ y el de la derecha es $\mathcal{N}$. Los cuatro cofactores ($\mathcal{M}_A$, $\mathcal{M}_{\overline{A}}$, $\mathcal{N}_A$, $\mathcal{N}_{\overline{A}}$) respecto de la variable del nodo raíz (*A*) han sido dibujados en la figura.

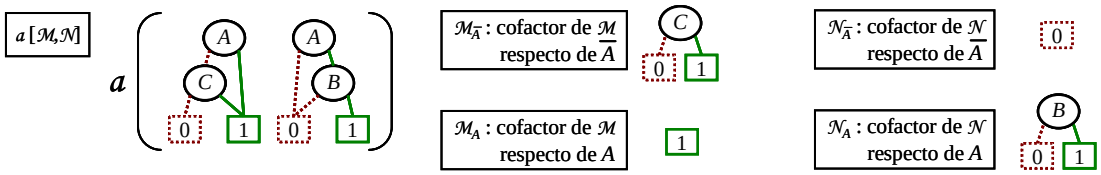


Figura 3-58: Cofactores de los operandos de la función de actividad respecto la variable del nodo raíz

La aplicación de la función de actividad sobre las cuatro combinaciones de cofactores se han dibujado en la figura 3-59. En la figura se han etiquetado estas cuatro combinaciones. Así, el 0→0 se corresponde con $a[\mathcal{M}_{\overline{A}}, \mathcal{N}_{\overline{A}}]$, esto es, el BDD resultante de la función de actividad de los cofactores de los BDD respecto a la variable negada del nodo raíz ($\overline{A}$). El 1→1, se corresponde con $a[\mathcal{M}_A, \mathcal{N}_A]$, esto es, el BDD resultante de la función de actividad de los cofactores de los BDD respecto a la variable del nodo raíz (A). El 1→0 se corresponde con $a[\mathcal{M}_A, \mathcal{N}_{\overline{A}}]$, esto es el BDD resultante de la función de actividad del cofactor del primer operando respecto a la variable del nodo raíz (A) y del cofactor del segundo operando respecto a la variable negada del nodo raíz ($\overline{A}$).

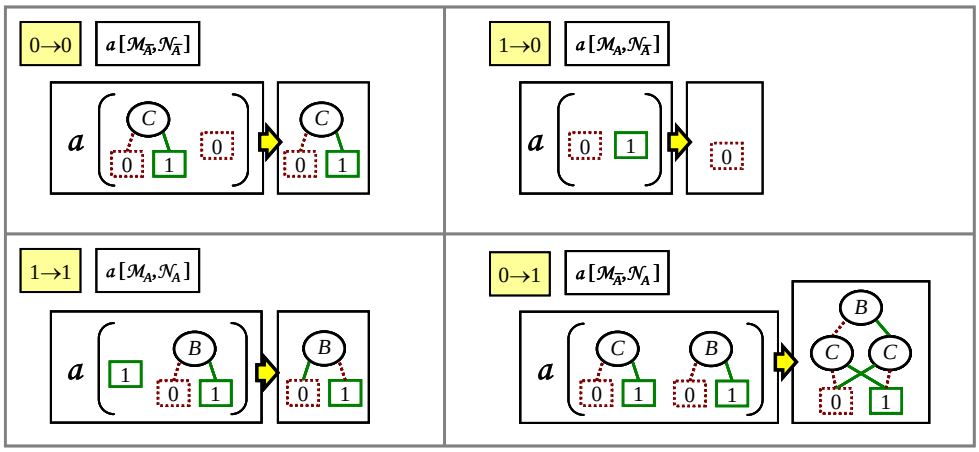


Figura 3-59: Resultados de la función de actividad aplicada a los cofactores

Estos cuatro BDD se colgarán de las cuatro ramas de la estructura *a*BDD creada para la variable A. Colgando de la rama 0→0 el BDD de la figura 3-59 con su misma etiqueta, e igual para la rama 1→1.

Aunque habitualmente los BDD de actividad (1→0 y 0→1) suelen ser iguales, no siempre ocurre así (como en el ejemplo de la figura 3-59). Cuando éstos son distintos existen dos formas de disponerlos, si se está creando un *a*BDD indiferenciado (§3.5.2.1) se cuelga el BDD más pequeño[41] de la rama cero de las ramas de actividad, lo que conduce a *a*BDD más compactos (§AII.4). En este caso se colgaría de la rama cero el BDD 1→0. En la figura 3-60 se muestra este proceso: de entre las dos ramas de actividad ($½a_A$) se escoge la rama cero para colgar el BDD correspondiente a 1→0, ya que es menor que el 0→1. Es habitual además, que uno de los dos BDD sea cero, lo que hace que se puedan simplificar con otras estructuras. El *ia*BDD resultante se ha dibujado a la derecha de la figura.

[41] En la herramienta BuDDy los BDD se definen por un valor único que normalmente tiene relación con su tamaño. En este caso, pequeño se refiere a este valor aunque pudiera ser de mayor tamaño.

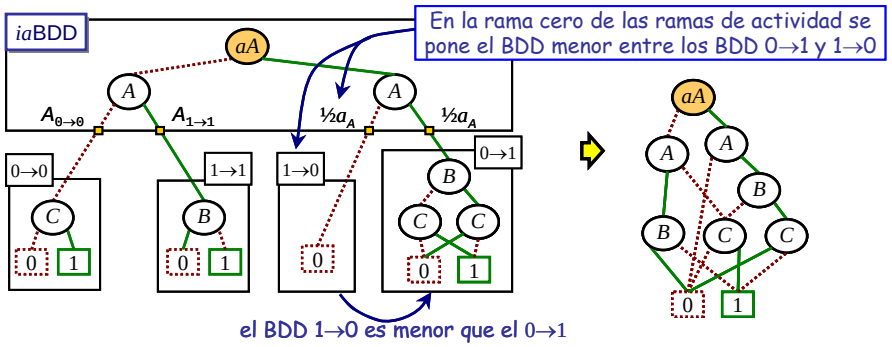


Figura 3-60: Obtención del iaBDD del ejemplo de la figura 3-58

En caso de que se quiera crear el *a*BDD diferenciado, siempre se pone en la rama cero el correspondiente al BDD 0→1. Para este ejemplo habría que colgar los BDD 0→1 y 1→0 justo de manera contraria (véase la figura 3-61). Para este ejemplo no hay diferencia de tamaño, y normalmente para un caso concreto no la hay. La reducción que logran los *ia*BDD se debe a que todas estas estructuras generales se crean de la misma manera, lo que da pie a que se reduzcan cuando desde otras ramas se tiene que crear la estructura (ver anexo AII.4).

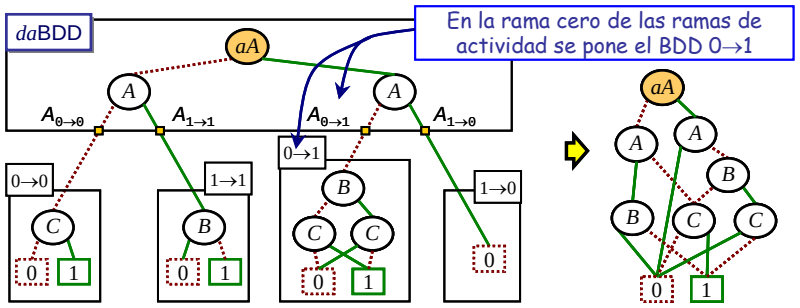


Figura 3-61: Obtención del daBDD del ejemplo de la figura 3-58

Debido a esta manera de disponer las ramas en los *da*BDD se puede observar que en la creación de éstos no se cumple la propiedad conmutativa, pues al cambiar el orden de los operandos de la figura 3-58, los BDD 0→1 y 1→0 saldrían al revés.

Para concluir con esta sección, se quiere resaltar la importancia de la definición del operador de actividad para la creación de los *a*BDD de manera independiente de los TFBDD, esto es, partiendo únicamente de los BDD de probabilidad. Todas las definiciones y propiedades descritas permiten la creación unívoca de los *a*BDD con las características definidas en el apartado 3.5.2.1. Esto además ha permitido automatizar el proceso e integrarlo en un programa informático que opera con BDD.

3.5.2.5 Obtención de la ecuación de actividad

La obtención de la ecuación de actividad no es un paso necesario para el cálculo de la actividad, ya que existen métodos más eficientes de calcular la actividad a partir de los BDD (ver anexo AI.2). Sin embargo, la obtención puede ser útil para entender los BDD y comparar los *a*BDD con los TFBDD.

Una vez que se ha creado el *a*BDD de una señal, a partir de él se obtiene su ecuación de actividad para poder calcular el valor numérico. La obtención de la ecuación de actividad a partir de un *a*BDD sigue un proceso similar al visto para la ecuación de probabilidad con los BDD (§2.2.2, figura 2-9) y la de la ecuación de actividad con los TFBDD (§3.5.1.2, figura 3-34).

Para obtener la ecuación a partir del *a*BDD se realiza el análisis de los caminos que llevan al nodo terminal uno, tal como se muestra en la figura 3-62.

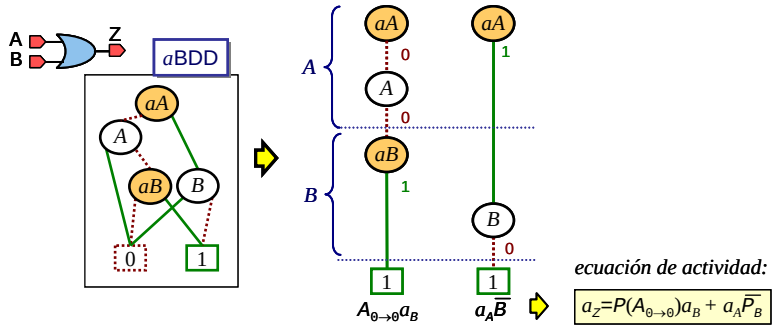


Figura 3-62: Obtención de la ecuación de actividad a partir de los caminos que llegan al nodo final 1 del aBDD de la puerta OR

Por tanto, de la figura 3-62 se extrae la ecuación de actividad de la puerta OR:

$$a_{OR} = a_A \cdot \overline{P_B} + P(A_{0\to0}) \cdot a_B \tag{3.16}$$

Mientras que para el TFBDD de la puerta OR se obtenía la siguiente ecuación, (explicada en la figura 2-25):

$$a_{OR} = P(A_{1\to0})\,\overline{P_B^T} + P(A_{0\to1})\,\overline{P_B^0} + P(A_{0\to0})\,P(B_{0\to1}) + P(A_{0\to0})\,P(B_{1\to0}) \tag{3.17}$$

Analizando las ecuaciones 3.16 y 3.17 se observa que son equivalentes, ya que al considerar que es un proceso estacionario (ecuación 2.2) se cumple que:

$$\overline{P_B^T} = \overline{P_B^0}$$

Y por tanto, la ecuación 3.17 se transforma en:

$$a_{OR} = [P(A_{1\to0}) + P(A_{0\to1})]\,\overline{P_B} + P(A_{0\to0})\,P(B_{0\to1}) + P(A_{0\to0})\,P(B_{1\to0})$$

Agrupando los dos últimos sumandos, y sabiendo que: $a_x=P(x_{1\to0})+P(x_{0\to1})$ (ecuación 2.5), el resultado es idéntico al obtenido mediante el *a*BDD (ecuación 3.16):

$$a_{OR} = a_A \cdot \overline{P_B} + P(A_{0\to0}) \cdot a_B$$

En consecuencia, **la representación en *a*BDD no supone una pérdida de exactitud en el cálculo**, sino que aprovecha las condiciones de *estacionariedad* y la igualdad de las probabilidades de transición de 0→1 y de 1→0 para construir una forma más compacta, en la que además, por estar basada en actividad e inactividad, se consiguen reducciones mayores.

3.5.3 Comparación entre TFBDD y *a*BDD de diversas funciones lógicas

En esta sección se comparan los tamaños de los BDD de actividad para distintas puertas y funciones lógicas según se representen en TFBDD o en *a*BDD.

En la figura 3-49 se mostró la diferencia entre el TFBDD y el *a*BDD para la puerta OR. Para la puerta NOR el resultado es idéntico, debido a que la actividad de una señal y su negada es la misma, ya que conmutan al mismo tiempo pero hacia valores opuestos.

En los siguientes subapartados se analizarán las diferencias entre los TFBDD y *a*BDD para las puertas AND, NAND, XOR, XNOR y para un multiplexor.

3.5.3.1 Puertas AND y NAND

Los BDD de actividad para las puertas AND y NAND son muy similares a los de la puerta OR. De hecho, la disposición de los nodos es la misma, y lo único que cambian son las ramas. Por tanto, como se puede apreciar en la figura 3-63, el TFBDD de la puerta AND tiene el mismo número de nodos que el TFBDD de la puerta OR; y lo mismo sucede para los *a*BDD. Así, igual

que para la puerta OR, se consiguen reducciones de casi un 43% en el número de nodos y de algo más de un 44% en el número de caminos.

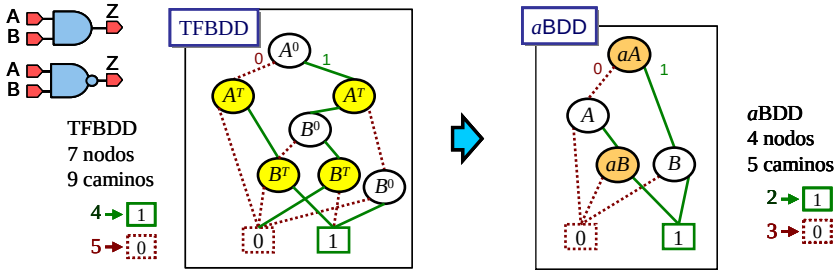


Figura 3-63: Comparación del TFBDD y aBDD para las puertas AND y NAND

3.5.3.2 Puertas XOR y XNOR

Para las puertas XOR y XNOR se consiguen formas muy compactas con los *a*BDD, esto se debe a que sus tablas de verdad están muy relacionadas con la actividad[42]. Así, como se puede apreciar en la figura 3-64, se pasa de 7 nodos con el TFBDD a 3 nodos con el *a*BDD, lo que supone una reducción de un 57% en el número de nodos. El número de caminos se reduce a la cuarta parte, pasando de 16 a tan sólo 4 caminos.

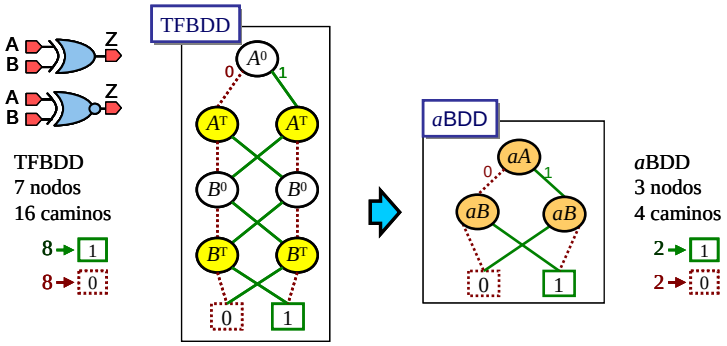


Figura 3-64: Comparación del TFBDD y aBDD para las puertas XOR y XNOR

3.5.3.3 Multiplexor

En un multiplexor también se consiguen importantes reducciones. Para evaluar la ventaja del uso de los *a*BDD respecto a los TFBDD, en la figura 3-65 se muestran ambas representaciones. Con el *a*BDD se consigue reducir de 13 a 7 nodos (46% menos) y reducir en la mitad el número de caminos (de 16 a 8).

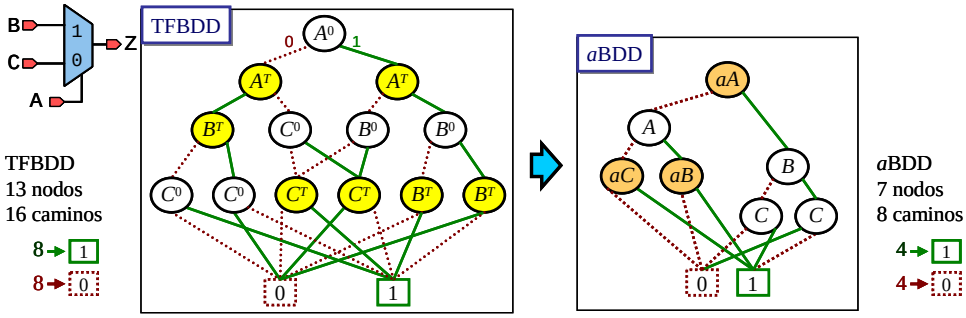


Figura 3-65: Comparación del TFBDD y aBDD para un multiplexor

[42] Obsérvese por ejemplo que la actividad se calcula aplicando la XOR en dos tiempos consecutivos de la señal

3.5.4 Tamaño de los BDD respecto al número de entradas

En esta sección se analizará el incremento del número de nodos y caminos de los BDD de probabilidad y actividad respecto al número de entradas de las puertas AND, XOR, y para los multiplexores.

3.5.4.1 Puerta AND

En la figura 3-66 se muestran los BDD resultantes para una puerta AND de tres entradas. El BDD de la función tiene 3 nodos y 4 caminos; mientras que el TFBDD tiene 11 nodos y 16 caminos, y el *a*BDD tiene 7 nodos y 9 caminos. Por tanto, la reducción de nodos del *a*BDD respecto del TFBDD es de un 36%, y la de los caminos es de un 44%. Estos datos se pueden contrastar con los de la AND de dos entradas (figura 3-63).

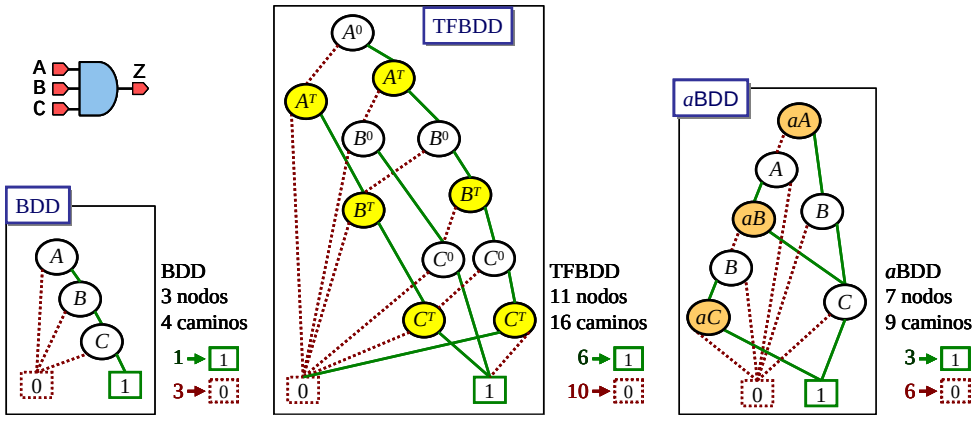


Figura 3-66: BDD, TFBDD y aBDD de una puerta AND de tres entradas

Los BDD resultantes de añadir una entrada más a la puerta AND se muestran en la figura 3-67. Ahora el número de nodos del TFBDD es 15 y el de *a*BDD es 10, por lo que hay una reducción del 33%. Mientras que el porcentaje de reducción del número de caminos del *a*BDD respecto del TFBDD es del 44%.

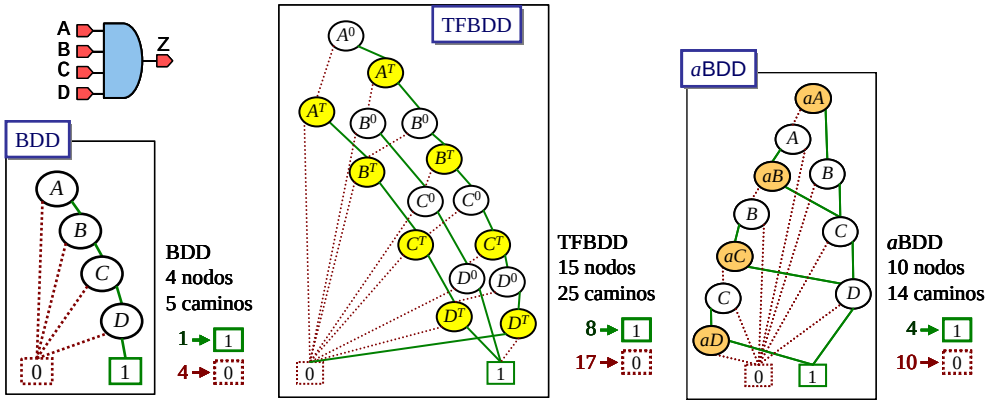


Figura 3-67: BDD, TFBDD y aBDD de una puerta AND de cuatro entradas

A partir de los ejemplos anteriores ya se podrían inferir los BDD resultantes de una puerta AND de cinco entradas. Éstos se han mostrado en la figura 3-68.

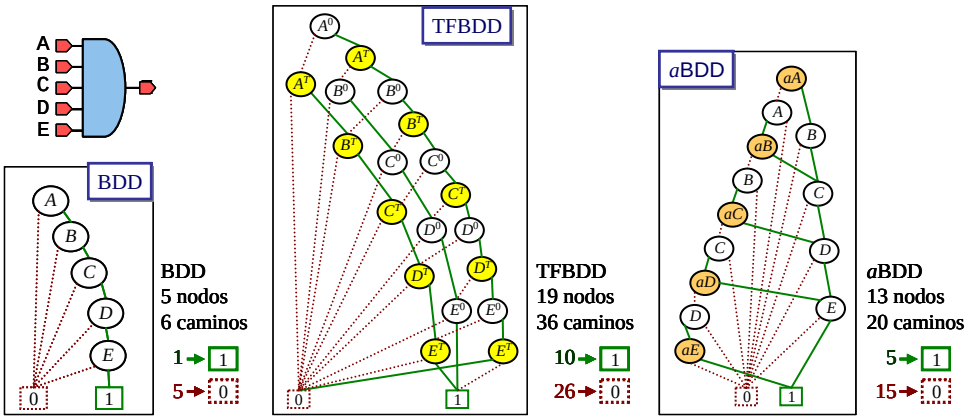


Figura 3-68: BDD, TFBDD y aBDD de una puerta AND de cinco entradas

En la tabla 3-1 se han incluido los resultados del número de nodos resultantes del BDD de probabilidad, del TFBDD y del *a*BDD, para una puerta AND según su número de entradas. En la última fila se ha obtenido el término general de las tres sucesiones. Como se puede ver, todas son progresiones aritméticas con diferente razón. El BDD tiene razón igual a la unidad, mientras que para el TFBDD su razón es cuatro y para el *a*BDD su razón es tres. Por tanto, cuando el número de entradas tiende a infinito, con el empleo de *a*BDD se consigue un ahorro de nodos que tiende al 25%. Como se ve en la última columna, la reducción de nodos es siempre igual o superior a al 25%.

Nº entradas de la AND	Número de nodos			reducción TFBDD→*a*BDD
	BDD	TFBDD	*a*BDD	
2	2	7	4	43%
3	3	11	7	36%
4	4	15	10	33%
5	5	19	13	32%
6	6	23	16	30%
n	n	4n-1	3n-2	$25\%|_{n\to\infty}$

Tabla 3-1: Número de nodos de los BDD de una AND según su número de entradas

La dependencia del número de caminos con las entradas de la AND se ha incluido en la tabla 3-2. Aquí, el incremento de número de caminos para la función de actividad es mayor, ya que para el BDD el incremento de caminos totales es lineal, mientras que para la actividad es cuadrática (TFBDD y *a*BDD). La reducción del número de caminos tiende a la mitad cuando el número de entradas tiende a infinito. Observando la tabla vemos que siempre se consiguen reducciones mayores de un 43%.

Nº de entradas	caminos 1			caminos 0			caminos totales			reduc. total TFBDD→*a*BDD
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	
2	1	4	2	2	5	3	3	9	5	44%
3	1	6	3	3	10	6	4	16	9	44%
4	1	8	4	4	17	10	5	25	14	44%
5	1	10	5	5	26	15	6	36	20	44%
6	1	12	6	6	37	21	7	49	27	45%
n	1	2n	n	n	n^2+1	$\frac{1}{2}(n^2+n)$	n+1	n^2+2n+1	$\frac{1}{2}(n^2+3n)$	$50\%|_{n\to\infty}$

Tabla 3-2: Número de caminos de los BDD de una AND según su número de entradas

Este análisis se puede trasladar con los mismos resultados de reducción de nodos y caminos[43] para las puertas NAND, OR y NOR.

3.5.4.2 Puerta XOR

A continuación se analizará el efecto de añadir entradas a la puerta XOR. La puerta XOR con múltiples entradas es equivalente a la separación de la operación en XOR de dos entradas, ya que la XOR tiene las propiedades conmutativa y asociativa. El resultado de la XOR de múltiples entradas es uno si tiene un número impar de entradas con valor uno, y si no vale cero. La figura E2-69 muestra la equivalencia de una puerta XOR de tres entradas y de otra de cuatro con puertas XOR de dos entradas, la extensión a un mayor número de entradas sigue la misma regla.

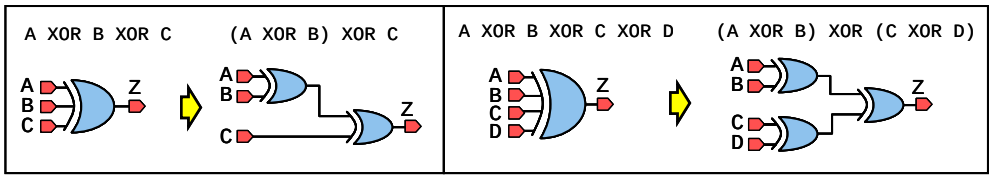


Figura 3-69: Equivalencia de una puerta XOR de tres y cuatro entradas

Los BDD de la puerta XOR de tres entradas se muestran en la figura 3-70. Se puede observar que el BDD y el *a*BDD tienen el mismo número de nodos y de caminos (5 nodos y 8 caminos), mientras que el TFBDD tiene más del doble de nodos y cuatro veces más el número de caminos (11 nodos y 32 caminos).

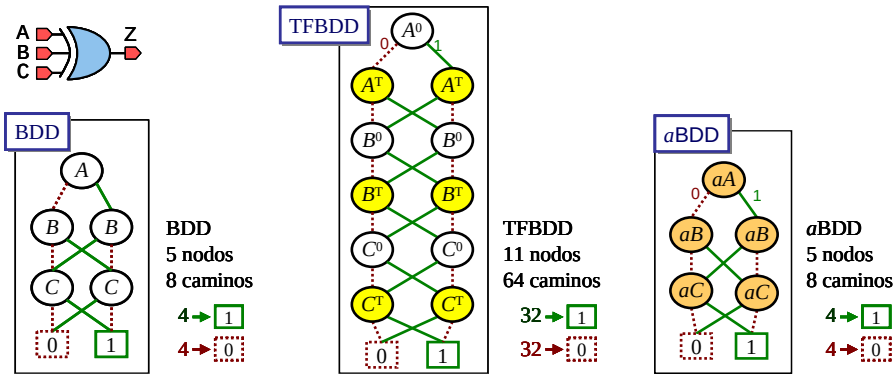


Figura 3-70: BDD, TFBDD y aBDD de una puerta XOR de tres entradas

Añadiéndole una cuarta entrada a la puerta XOR, sus BDD quedan como los mostrados en la figura 3-71. El *a*BDD se mantiene con el mismo número de nodos y caminos que el BDD (7 nodos y 16 caminos), mientras que se puede apreciar el rápido crecimiento de los TFBDD (15 nodos), en especial el referente al número de caminos (256).

[43] Aunque las cifras referentes a los caminos a cero y a uno pueden estar intercambiadas

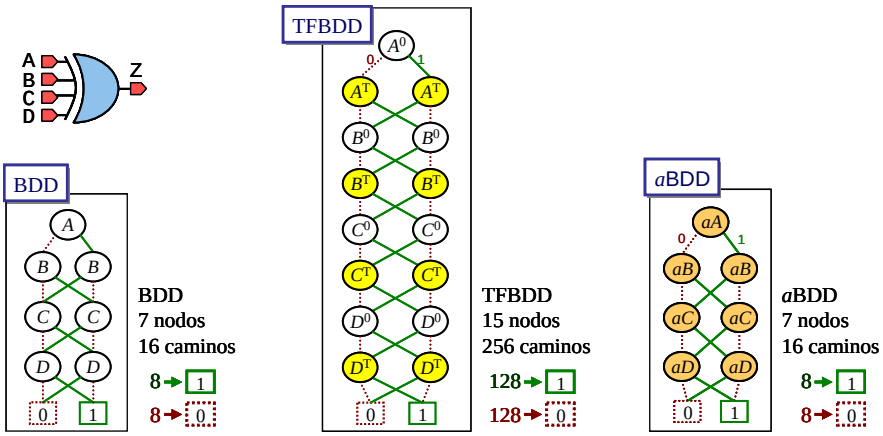


Figura 3-71: BDD, TFBDD y aBDD de una puerta XOR de cuatro entradas

Se ha realizado una tabla con la evolución del tamaño de los BDD en función del número de entradas de la XOR (tabla 3-3). El incremento de nodos para los BDD y los *a*BDD sigue una progresión aritmética de razón dos, mientras que el número de nodos para el TFBDD sigue una progresión aritmética de razón cuatro. Por tanto, cuando el número de entradas se hace muy grande, la reducción de nodos por representar la actividad mediante *a*BDD en vez de TFBDD tiende al 50%. Además, el hecho de lograr una representación que no hace aumentar el número de nodos para los BDD de actividad respecto a los de probabilidad es muy interesante.

Nº entradas de la XOR	Número de nodos			reducción TFBDD→*a*BDD
	BDD	TFBDD	*a*BDD	
2	3	7	3	57%
3	5	11	5	54%
4	7	15	7	53%
5	9	19	9	53%
6	11	23	11	52%
n	2n-1	4n-1	2n-1	$50\%\|_{n\to\infty}$

Tabla 3-3: Número de nodos de los BDD de una XOR según su número de entradas

La progresión que sigue el número de caminos de los BDD de la XOR según su número de entradas es geométrica, es por tanto un crecimiento exponencial. Sin embargo, mientras que la razón de la progresión para el BDD y el *a*BDD es dos, la razón para el TFBDD es cuatro. Por tanto, como se puede apreciar en la tabla 3-4, el aumento del número de caminos para los TFBDD es mucho más rápido que en los otros dos.

Nº de entradas	caminos 1			caminos 0			caminos totales			reduc. total TFBDD→*a*BDD
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	
2	2	8	2	2	8	2	4	16	4	75%
3	4	32	4	4	32	4	8	64	8	87%
4	8	128	8	8	128	8	16	256	16	94%
5	16	512	16	16	512	16	32	1024	32	97%
6	32	2048	32	32	2048	32	64	4096	64	98%
n	2^{n-1}	2^{2n-1}	2^{n-1}	2^{n-1}	2^{2n-1}	2^{n-1}	2^n	2^{2n}	2^n	$100\%\|_{n\to\infty}$

Tabla 3-4: Número de caminos de los BDD de una XOR según su número de entradas

3.5.4.3 Multiplexor

Para realizar el análisis del tamaño de los BDD de un multiplexor según su número de entradas, **no se va a dividir el circuito por el método de regiones disjuntas** (§3.4.1), y así **hacer un análisis independiente del método de partición escogido**.

A diferencia de los análisis anteriores, en éste se va a distinguir entre los *a*BDD diferenciados (*da*BDD) y los indiferenciados (*ia*BDD). Éstos fueron explicados en la sección 3.5.2 y en el anexo AII.4. Como se ha indicado, en todos los ejemplos anteriores no ha sido necesaria la distinción entre ellos. En todo caso, el *a*BDD indiferenciado es el empleado si no se especifica lo contrario.

Sin embargo, como se verá en este apartado, en los multiplexores existen notables diferencias de tamaño. El análisis ayudará a comprender ambos BDD.

El BDD de la probabilidad de un multiplexor de dos alternativas (y una condición) se mostró en la figura 3-33. El TFBDD y el *a*BDD se compararon en la sección 3.5.3.3. Para este multiplexor, no hay diferencias entre su *ia*BDD y su *da*BDD.

Si se añade una alternativa al multiplexor se tiene que añadir también una nueva señal de selección. Este multiplexor no es completo, existiendo una condición que no asigna un valor determinado, o bien una alternativa es asignada en dos condiciones.

La figura 3-72 muestra los BDD para un multiplexor de tres alternativas. La alternativa *D* se selecciona cuando la señal de selección *A*1 vale uno, independientemente del valor de la otra señal de selección (*A*0). De la figura se puede observar el gran aumento de nodos y caminos de ambos BDD de actividad respecto al BDD de probabilidad. De los 5 nodos que tiene el BDD de probabilidad, se pasa a 27 para el TFBDD y 15 para el *a*BDD. Aún así, la diferencia entre el *a*BDD y el TFBDD es considerable, logrando una reducción del 44% en el número de nodos, y del 50% en el número de caminos. Para este caso, el *a*BDD diferenciado e indiferenciado son iguales, y por tanto, se han representado juntos.

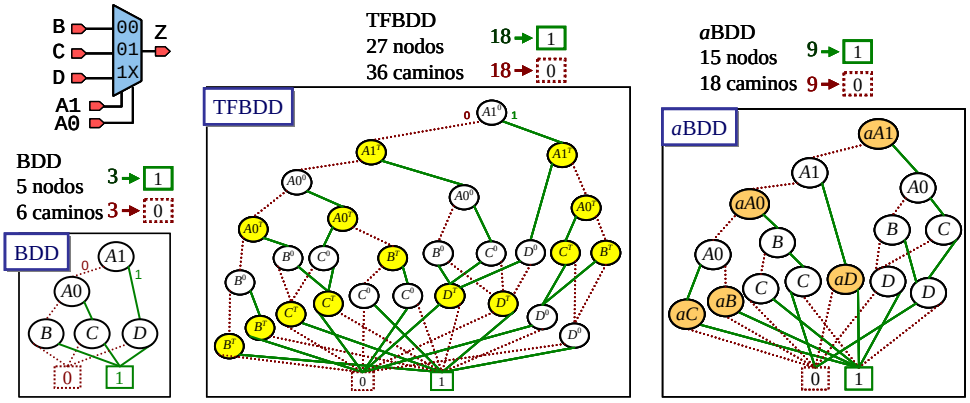


Figura 3-72: BDD, TFBDD y aBDD de un multiplexor de tres alternativas

Con cuatro alternativas se queda un multiplexor completo, en el que no hay alternativas que se seleccionen con más de una condición. Aquí se puede notar el rápido incremento de los BDD de actividad. La figura 3-73 muestra el BDD de probabilidad y el TFBDD. En la figura E2-74 se han dibujado los *a*BDD para este multiplexor. Mientras que el BDD de probabilidad tan sólo tiene 7 nodos, el TFBDD tiene 45 nodos; el *ai*BDD 25 nodos y el *di*BDD 28 nodos. Por otro lado, los caminos del TFBDD son 64, justo el cuadrado de los caminos del BDD de probabilidad. El *ia*BDD tiene la mitad de caminos que el TFBDD, y el *da*BDD tiene un 50% de caminos más que el *ia*BDD.

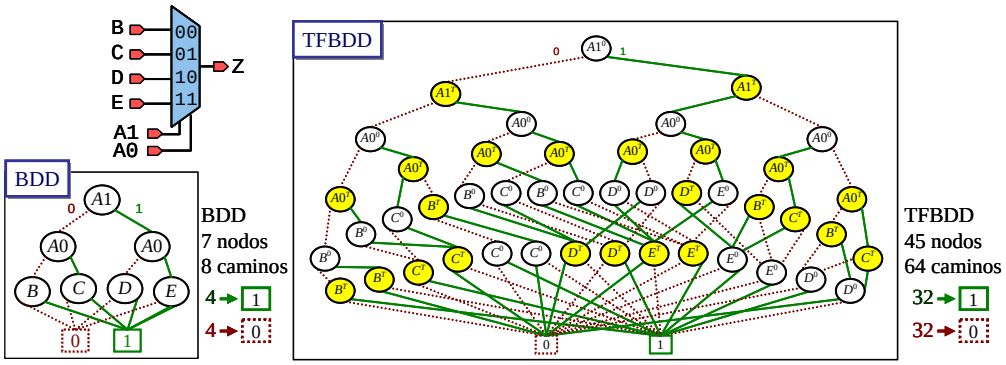


Figura 3-73: BDD y TFBDD de un multiplexor de cuatro alternativas

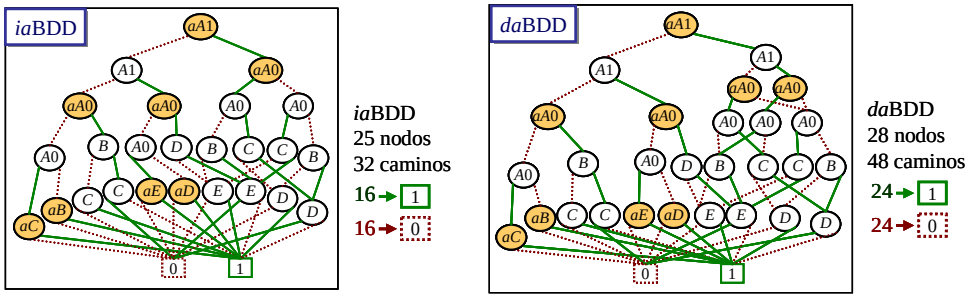


Figura 3-74: BDD y TFBDD de un multiplexor de cuatro alternativas

El tamaño de los multiplexores de más de cuatro alternativas que no son completos[44] depende de cómo se distribuyen las asignaciones de las alternativas, y por lo tanto, no hay un único tamaño del BDD para un número de alternativas. Por ejemplo, en la figura 3-75 se muestran dos multiplexores de 6 alternativas en los que las alternativas *F* y *G* se distribuyen de manera diferente. Los BDD del multiplexor de la izquierda (mux6-d1) son menores porque su distribución es más homogénea.

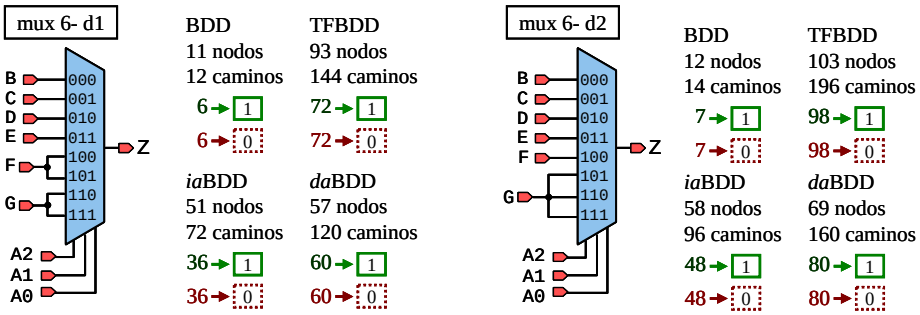


Figura 3-75: Tamaños de los BDD para dos configuraciones de un multiplexor de 6 alternativas

Debido a estas variaciones resulta difícil extraer una secuencia a partir de los datos de crecimiento de los BDD con el número de alternativas. En la tabla 3-5 se ha puesto el número de nodos de los BDD para distinto número de alternativas del multiplexor, a partir del multiplexor de 8 alternativas sólo se han puesto multiplexores completos (16, 32, 64, 128 y 256 alternativas) ya que éstos no dependen de la distribución de la asignación de alternativas. Se han incluido también dos multiplexores posibles para el de 6 alternativas, cuya estructura se ha representado en la figura 3-75.

La reducción de nodos obtenida mediante el uso de *ia*BDD frente a los TFBDD tiende al 50%, con los *da*BDD el porcentaje es menor y baja cuanto mayor es el multiplexor: 31% de reducción para el multiplexor de 16 alternativas.

[44] Que no tengan un número de alternativas que sea potencia de dos

Alternativas multiplexor	BDD	TFBDD	*ia*BDD	*da*BDD	reducción TFBDD→*ia*BDD	reducción TFBDD→*da*BDD
2	3	13	7	7	46%	46%
3	5	27	15	15	44%	44%
4	7	45	25	28	44%	38%
5	9	67	37	40	45%	40%
6 -d1	11	93	51	57	45%	39%
6 -d2	12	103	58	69	44%	33%
7	13	123	67	78	46%	37%
8	15	157	85	103	46%	34%
16	30	530	283	367	47%	31%
32	63	2173	1117	1477	49%	32%
64	127	8445	4285	5773	49%	32%
128	255	33277	16765	22813	50%	31%
256	511	132093	66301	90685	50%	31%
n	2n-1	$\sim 2(n+1)^2$	$\sim (n+1)^2$	$\sim (4/3)(n+1)^2$	$50\%\|_{n\to\infty}$	$31\%\|_{n\to\infty}$

Tabla 3-5: Número de nodos de los BDD de un multiplexor según su número de alternativas

Para analizar el número de caminos se pondrán la suma de los caminos totales que van a los nodos finales cero y uno, ya que para todos los casos, son iguales (por tanto basta con dividir por dos). Estos datos se han incluido en la tabla 3-6, en donde se puede apreciar que el número de caminos de los *ia*BDD tiende a la mitad del número de caminos del TFBDD.

Nº de alternativas	caminos totales				reducción TFBDD→*ia*BDD	reducción TFBDD→*da*BDD
	BDD	TFBDD	*ia*BDD	*da*BDD		
2	4	16	8	8	75%	75%
3	6	36	18	18	50%	50%
4	8	64	32	48	50%	25%
5	10	100	50	66	50%	34%
6 -d1	12	144	72	120	50%	17%
6 -d2	14	196	96	160	50%	18%
7	14	196	98	162	50%	17%
8	16	256	128	224	50%	12%
16	32	1024	512	956	50%	7%
32	64	4096	2048	3968	48%	3%
64	128	16384	8192	16128	49%	2%
128	256	65536	32768	65024	50%	1%
256	512	262144	131072	261129	50%	0%
n	2n	$4n^2$	$2n^2$	$4n^2\|_{n\to\infty}$	$50\%\|_{n\to\infty}$	$0\%\|_{n\to\infty}$

Tabla 3-6: Número de caminos totales de los BDD de un multiplexor según su número de alternativas

Como conclusión de este apartado se señala la **importante reducción del número de nodos que se logran con los *a*BDD frente a los TFBDD**, yendo desde un 25% para puertas AND, OR grandes, hasta el 50% en multiplexores y puertas XOR. En el número de caminos se logran reducciones aún mayores.

3.6 Ordenamiento RTL de los BDD

El orden de las variables de los BDD tiene una influencia decisiva en su tamaño [13], [52], [78], [112]. Así, por ejemplo, según el orden de variables escogido, el tamaño del BDD de un sumador de n bits puede variar entre $O(n)$ y $O(2^{n/2})$ [78]. Es por esto que se hace necesario escoger un orden adecuado para construir el BDD. Sin embargo, la determinación del orden óptimo es un problema NP-Hard[45] sobre el que se ha investigado en abundancia [52].

Para un circuito con n entradas existen $n!$ órdenes posibles, pues se trata de permutaciones de n elementos. Consecuentemente, para circuitos grandes, no sólo el tamaño de los BDD resultantes puede ser un problema, sino que también la elección del orden apropiado.

En esta tesis se propone aprovechar la información disponible desde el nivel RT para escoger un orden favorable. Habitualmente, desde el nivel RT se disponen las señales con un cierto orden. Considerar el *orden natural* del nivel RT consistiría en seguir el orden de aparición de las señales al recorrer el modelo extendido del hardware (EHM, §3.3.2). Además, desde el nivel RT se cuenta con la información de las señales agrupadas en vectores y de los operadores involucrados. Esta información permite **ordenar adecuadamente los índices de los vectores** según las operaciones en las que se encuentren.

Estas capacidades de ordenación: la relativa a la disposición de las señales en RTL y la relativa a la información de los operadores y los índices de los vectores, **no son posibles de obtener de manera fácil desde el nivel de puertas**.

A continuación se darán las directrices para obtener el orden *natural* del RTL. Estas directrices han sido obtenidas de manera empírica. Por ello, en el apartado se verificará su validez mediante ejemplos sencillos, y en el capítulo 4 se corroborarán con ejemplos de mayor tamaño. El orden relativo a los índices de los vectores y los operadores se verá de manera práctica en el capítulo de resultados (capítulo 4).

Para obtener el orden RTL hay que seguir ciertas **reglas** que son fáciles de cumplir al recorrer el EHM:

1. Las señales de las condiciones de mayor jerarquía (sentencias condicionales externas) van antes que las señales contenidas en las sentencias afectadas por dichas condiciones (ver ejemplo 1, figura 3-77)
2. Por tanto, a las condiciones de mayor jerarquía le siguen las siguientes condiciones hasta llegar a las asignaciones que están afectadas por la condición
3. Una vez que se ha tomado una rama, se recorren antes las ramas inferiores hasta cubrir todo el sub-árbol antes de pasar a otras ramas y sub-árboles
4. Por tanto, una señal de una sentencia de asignación puede estar antes que una condición si no está afectada por ella
5. Las señales reconvergentes se pondrán en último lugar dentro de su región de reconvergencia (ver ejemplo 2, figura 3-79)
6. En general, en el EHM se recorren antes las alternativas de un multiplexor que tengan más complejidad, profundidad del EHM o número de dependencias
7. Y lo mismo para una puerta lógica, se recorre antes la entrada que tenga la señal de mayor complejidad, profundidad o número de dependencias (ver ejemplo 5, figura 3-85)
8. Cuando se toman las reglas 6 ó 7 las señales reconvergentes que afectan a ambas alternativas o entradas se pueden poner al final de la alternativa compleja tomada, o ponerse al final de toda su área de reconvergencia (ver ejemplo 6 y 8)
9. También se puede tomar antes la alternativa o la señal con menor profundidad, pero si hay mucha diferencia de número de dependencias finales entre las alternativas, las

[45] Problemas NP-Hard [45], [102]

señales de reconvergencia no se ponen al final de la zona de reconvergencia, sino donde aparezcan (ver ejemplos 5, figura 3-84)

Puede observarse que estas reglas son abiertas, es decir, ofrecen la posibilidad de varios tipos de ordenación. Para entender estas reglas es mejor hacerlo a través de los ejemplos que se verán en las secciones siguientes. Como se podrá apreciar, **mediante el uso del EHM el procedimiento es sencillo**, por el contrario, **establecer este orden no es fácil desde el nivel de puertas** ya que no se tiene el conocimiento de las señales que son condiciones de multiplexores y normalmente se escoge un orden aleatorio al que posteriormente se le podrán aplicar algoritmos de ordenación.

Estas implicaciones se han inferido a partir de la observación, sin que se haya desarrollado una teoría que fundamente esta hipótesis. Aún así, el planteamiento tiene lógica, ya que, como se verá en el siguiente apartado, las reglas se acoplan bien a la estructura del BDD. En el apartado siguiente se muestran diversos ejemplos en donde aplican las reglas de ordenamiento y se compara el tamaño de los BDD resultantes con todos los órdenes posibles.

3.6.1 Ejemplos de ordenación RTL

A continuación se mostrarán varios ejemplos en los que se observa el fenómeno descrito y que además servirán para ilustrar el proceso de la elección del orden RTL. En estos ejemplos **no se realizará la partición del circuito en regiones disjuntas** para comparar los BDD resultantes relativos a las mismas áreas.

3.6.1.1 Ejemplo 1

En el primer ejemplo se verá una sentencia condicional simple que se corresponde con un multiplexor. Si se toma la señal de la condición en primer lugar, la estructura del BDD es una representación de la sentencia condicional. Esto es, tal como se muestra en la figura 3-76, al tomar el orden de variables *ABC* el nodo raíz *A* selecciona la alternativa *B* ó *C* según su valor (si vale 1 selecciona *B* y si vale 0 selecciona *C*). Sin embargo al tomar cualquier otra variable como la primera, se evalúa dicha variable sin saber si la condición la ha seleccionado, lo que habitualmente produce un mayor número de nodos.

Para este ejemplo, el peor caso es cuando la condición está al final (BDD de la derecha de la figura) en cualquiera de los dos órdenes posibles (*CBA* ó *BCA*). Para esta situación se obtiene un BDD de 5 nodos frente a los 3 nodos del BDD con la condición en primer lugar, lo que supone una reducción de 40% en el número de nodos.

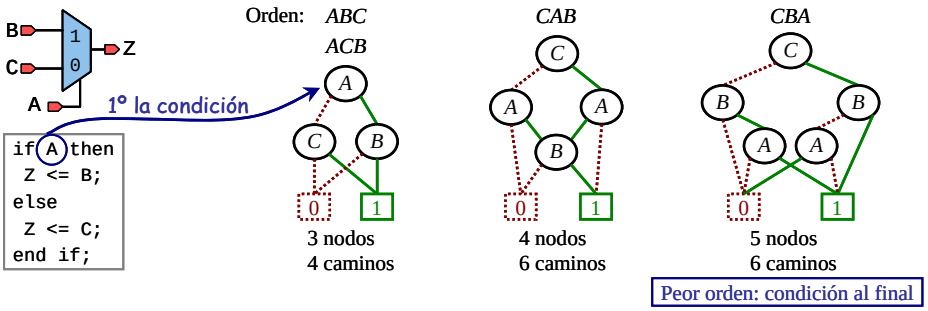


Figura 3-76: Tamaño de los BDD de un multiplexor según el orden de las variables

Para entender la elección del orden, en la figura 3-77 se ha dibujado el EHM y los recorridos para su obtención. Partiendo de la señal de salida (*Z*) se llega a la sentencia condicional (representada por un rombo). Al llegar a la sentencia condicional primero se debe ir a la condición (regla 1), con lo que la señal *A* va a ser la primera. Posteriormente hay dos opciones, ir por el camino cero (línea discontinua: recorrido de la izquierda en la figura) o por el camino uno

(derecha). Para este ejemplo, cualquiera de los dos caminos proporciona un BDD mínimo, cuyos órdenes son *ACB* y *ABC*.

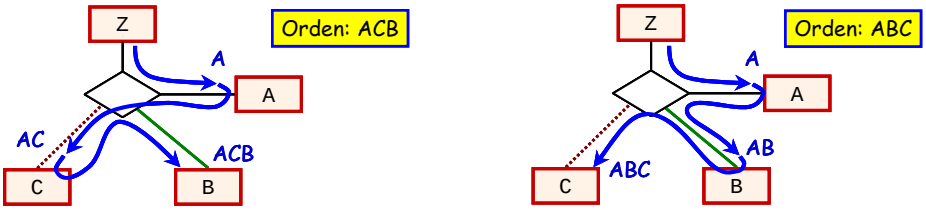


Figura 3-77: Recorrido del EHM para la obtención del orden de las variables del BDD de un multiplexor

El orden de las variables también afecta a los BDD de actividad, un buen orden para el BDD de probabilidad también resulta un buen orden para el BDD de actividad. En la tabla 3-7 se indica el número de nodos y caminos según el orden para los BDD de probabilidad, los TFBDD y los *a*BDD.

Nótese cómo **combinando el uso de los *a*BDD con el orden natural del procesamiento RTL** se consigue pasar de 21 nodos y 36 caminos en el peor caso y haciendo uso de los TFBDD, a tan sólo 7 nodos y 8 caminos. Lográndose una **reducción del número de nodos a la tercera parte** y el número de caminos a más de más de la cuarta parte.

Orden	Nodos			Caminos		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
ABC - ACB	3	13	7	4	16	8
BAC - CAB	4	18	11	6	36	15
BCA - CBA	5	21	12	6	36	17

Tabla 3-7: Tamaño de los BDD de un multiplexor según el orden de sus variables

3.6.1.2 Ejemplo 2

A continuación se mostrará la variación de tamaño de los BDD para un circuito mayor, éste se ha dibujado en la figura 3-78 (es el mismo circuito de la figura 3-18). Como se ha comentado, para hacer este análisis no se realiza la partición por regiones disjuntas propuesta en la sección 3.4, ya que si se realizase, el área que se analizaría sería un multiplexor sencillo, como el del ejemplo anterior (figura 3-76).

Debido a que el circuito tiene 4 entradas, tiene 24 posibles órdenes de variables para el BDD. Para dar una idea del aspecto y la variación de estos BDD en la figura se han mostrado cuatro de ellos. En la figura se ha dibujado uno de los BDD con menor tamaño (5 nodos), en el cual, la condición (*A*) está la primera y la señal reconvergente (*C*) en el último lugar. Por tanto, este BDD es el que se construiría siguiendo las reglas del orden RTL. Los peores casos tienen la condición y la señal reconvergente en el último lugar, que es el BDD de la derecha de la figura, con 9 nodos y 12 caminos.

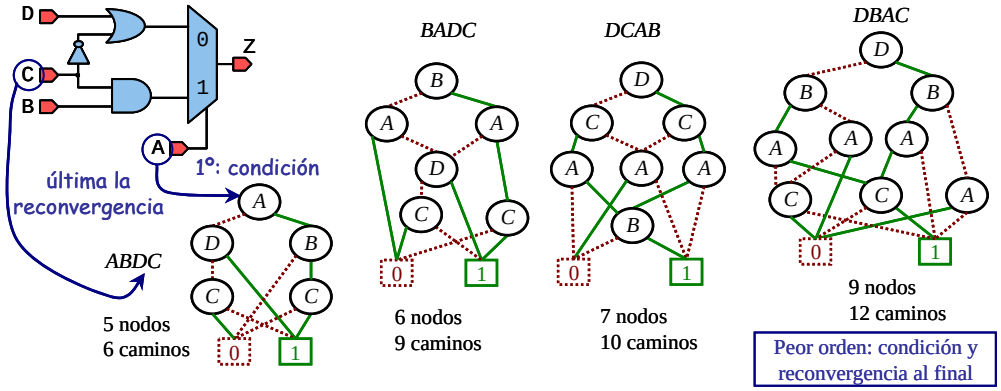


Figura 3-78: Tamaño de varios BDD del ejemplo 2 según el orden de las variables

La obtención del orden de las variables se logra recorriendo el EHM como se muestra en la figura 3-79. Esto es, partiendo de la señal que se está estudiando (*Z*) se llega a la sentencia condicional (rombo), de ella siempre se va a la expresión de la condición, en donde se toma la señal *A* como la primera en el orden. Posteriormente se puede ir por uno de los dos caminos, el de la rama cero (línea de puntos, a la izquierda) o la rama uno (línea discontinua, a la derecha). Como ambos caminos tienen el mismo número de dependencias finales se puede tomar primero cualquiera de ellos y se debe dejar la señal de reconvergencia para el final (regla 5).

El recorrido de la izquierda toma primero la rama cero, de ella se obtiene la señal *D* como segunda señal en el orden, mientras que al pasar por la señal *C* se deja para el final por ser reconvergente. Posteriormente, al ir por la rama uno se añade la señal *B*, y por último se incluye al final la señal reconvergente *C*. El orden obtenido es por tanto *ADBC*.

De manera similar, pero tomando antes la rama uno que la rama cero, se obtiene el orden *ABDC*, que se muestra en el dibujo de la derecha de la figura 3-79.

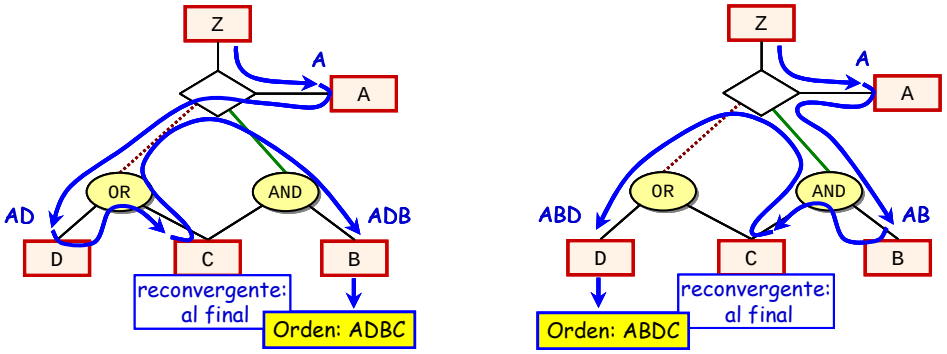


Figura 3-79: Recorrido del EHM para la obtención del orden de las variables del BDD del ejemplo 2

En la tabla 3-8 se han ordenado los tamaños de los BDD para las 24 permutaciones de las variables. Como se ha dicho, el orden RTL es el que tiene la condición en primer lugar y la señal reconvergente en último, teniendo las variables *B* y *D* en cualquier posición entre ellas. Esta forma de ordenar el BDD resulta la óptima.

Orden	Nodos			Caminos		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
ABDC - ADBC	5	21	13	6	36	17
CABD - CADB	5	27	15	6	36	18
ABCD - ADCB	5	25	16	6	36	19
ACBD - ACDB	5	27	16	6	36	19
BADC - DABC	6	26	17	9	81	34
CBDA - CDBA	6	28	17	8	64	25
BACD - DACB	6	30	20	9	81	38
CBAD - CDAB	6	31	21	8	64	31
BCDA - DCBA	7	33	21	10	100	37
BCAD - DCAB	7	36	25	10	100	47
BDCA - DBCA	8	45	27	12	144	51
BDAC - DBAC	9	53	32	12	144	53
Media	6,3	31,8	20	8,5	76,8	32,4
Mediana	6	29	18,5	8,5	72,5	32,5

Tabla 3-8: Tamaño de los BDD del ejemplo de la figura 3-78 según el orden de sus variables

Nótese la diferencia de haber escogido el orden de la última fila (*BDAC* ó *DBAC*) y usar TFBDD, respecto a usar el orden RTL (primera fila) y usar los *a*BDD. Se pasa de 53 nodos a 13, que es más de la cuarta parte.

Si el análisis se hubiese hecho desde el nivel de puertas el orden se hubiese tomado de manera aleatoria, y se puede suponer que el orden escogido no sería ni el óptimo ni el peor, sino un orden intermedio, por ejemplo el correspondiente a la mediana. Comparando estos valores se pasaría de 29 nodos usando TFBDD (ó 18-19 con *a*BDD) a 13 nodos con *a*BDD y con el orden RTL. Esto supone una reducción de más de un 50% en el número de nodos.

Es interesante además ver la reducción en el número de nodos de los *a*BDD frente a los TFBDD, dado un mismo orden. Esta reducción va desde el 30,6% para la fila novena (*BCAD*, *DCAB*), hasta el 44,4% para la segunda fila (*CABD*, *CADB*).

3.6.1.3 Ejemplo 3

Ahora se va a estudiar un ejemplo con dos sentencias condicionales. Este ejemplo se corresponde con la zona de reconvergencia del ejemplo de la figura 3-20. De la misma manera que en el ejemplo anterior, no se va a realizar la partición por regiones disjuntas para que la región no se quede como un multiplexor sencillo.

En la figura 3-80 se muestra el circuito y tres BDD con distintos órdenes de las 120 posibilidades de ordenación (pues son 5 variables). Entre ellos está un BDD con tamaño mínimo (6 nodos), otro con tamaño medio (10 nodos) y el de mayor tamaño (15 nodos). Se puede observar que el tamaño mínimo se logra poniendo la condición principal (*A*) en primer lugar, y poniendo la condición *M* antes de sus alternativas (*D* y *E*).

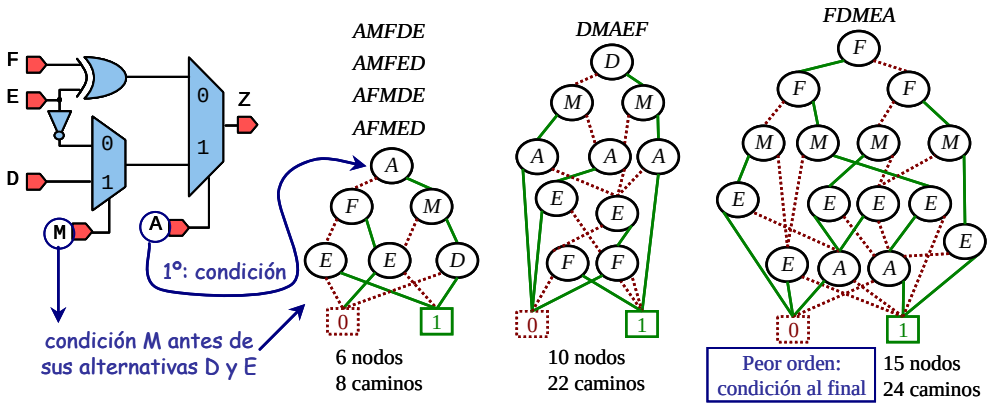


Figura 3-80: Tamaño de varios BDD del circuito según el orden de las variables

Los dos recorridos para obtener el orden RTL están mostrados en la figura 3-81. Como la complejidad de las alternativas es similar, teniendo ambas un número parecido de dependencias finales, se puede recorrer en primer lugar cualquiera de ellas, colocando la señal reconvergente al final (regla 5). La interpretación es similar al ejemplo anterior: al pasar por las sentencias condicionales se va primero a la expresión de la condición, y se dejan las señales reconvergentes para el final. Siguiendo estos recorridos se obtienen los órdenes *AMFDE* y *AFMDE*.

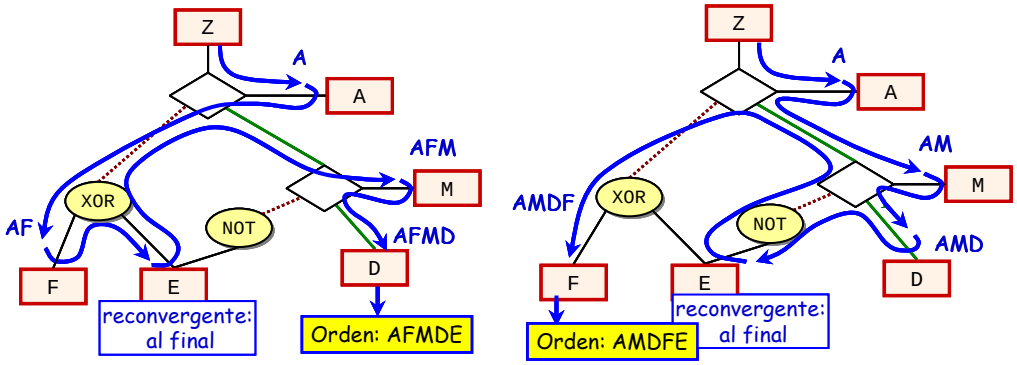


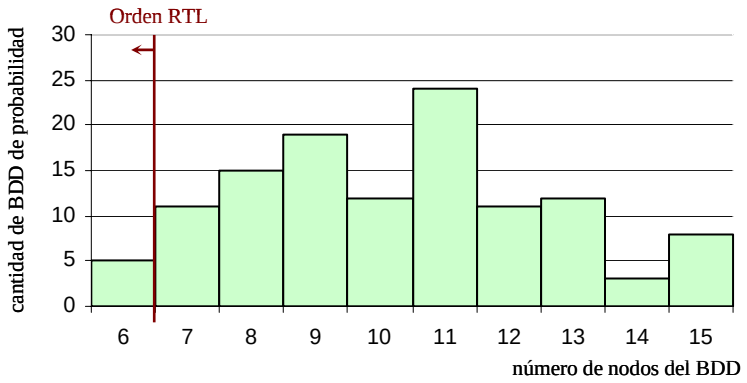
Figura 3-81: Recorrido del EHM para la obtención del orden de las variables del BDD del ejemplo 3

En la tabla 3-9 se han puesto algunos de los órdenes de variables con su número de nodos y caminos de los BDD resultantes. Para no hacer una tabla muy extensa se ha limitado el número de filas, poniendo únicamente todos los órdenes de variables que dan BDD de 6 y 7 nodos, y un sólo orden (o varios si tienen resultados coincidentes) para los BDD que tienen de 8 a 15 nodos. Los órdenes RTL se han señalado en negrita, estando uno en la primera y el otro en la segunda fila. Las cuatro últimas filas de la tabla incluyen la media, la mediana, el primer cuartil y el primer decil considerando la distribución completa y no sólo los datos que figuran en la tabla. Puede observarse que los dos órdenes resultantes del método propuesto están por debajo del primer decil.

Orden	Nodos			Caminos		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMDE *- AFMED -AMFDE - AMFED*	6	27	15	8	62	24
AMDFE	6	29	16	8	64	24
FAMDE - FAMED - MAFDE - MAFED	7	32	19	12	144	54
AMDEF	7	37	19	8	64	24
AMEDF - AMEFD	7	39	19	8	64	24
MADEF	7	34	20	12	144	44
ADMFE	7	36	21	10	100	37
ADFME - AFDME	7	37	22	10	100	37
MAEDF - MAEFD	8	44	23	12	144	44
AFEDM	9	47	27	10	100	43
EMADF - EMAFD	10	60	40	14	196	83
EDAFM - EDAMF	11	51	32	14	196	69
FMDAE - MDFAE - MFDAE	12	75	44	18	324	109
DFEMA - EFDMA - FDEMA - FEDMA	13	93	53	18	324	116
DFMAE - FDMAE	14	95	59	24	576	188
DFMEA - FDMEA	15	105	67	24	576	196
[...]						
Media	10,27	60,13	36,05	15,23	250,7	89,65
Mediana	10	51,5	32	16	256	90
1er cuartil	8	46	27	11,5	133	45
1er decil	7	37	19,9	10	100	42,4

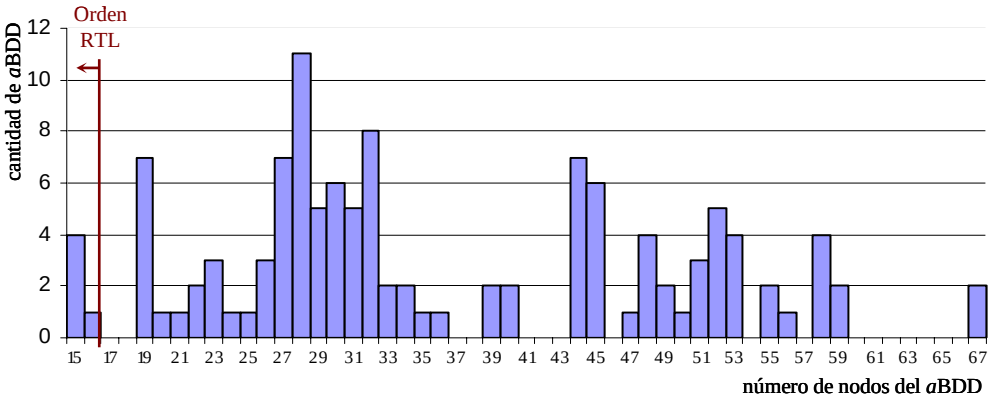
Tabla 3-9: Tamaño de algunos los BDD del ejemplo de la figura 3-80 según el orden de sus variables

Como la tabla 3-9 no es completa, se ha representado el histograma de los 120 BDD de probabilidad en la gráfica 3-1. En las abscisas figura el número de nodos, que van de 6 a 15, y la altura de la barra representa la cantidad de BDD encontrados con ese número de nodos. Los BDD obtenidos a través del orden RTL se encuentran en la primera barra (6 nodos).



Gráfica 3-1: Distribución de los BDD de probabilidad según su número de nodos

También se ha dibujado la distribución de los *a*BDD según su número de nodos en la gráfica 3-2. En dicha gráfica, los órdenes de los BDD propuestos por el RTL están en las dos primeras barras (15 y 16 nodos).



Gráfica 3-2: Distribución de los aBDD según su número de nodos

Comparando el peor caso en TFBDD con los órdenes RTL propuestos usando *a*BDD se pasa de 105 nodos a 15 ó 16, lo que es una diferencia bastante notable. Si en ambos casos se usan *a*BDD se pasa de 67 a 15 nodos. Comparando con el valor de la mediana, se pasa de 51 nodos en TFBDD ó 32 en *a*BDD a los 15 ó 16 nodos ya comentados.

3.6.1.4 Ejemplo 4

Ahora se tratará un circuito con más de una señal de reconvergencia. En la figura 3-82 se muestra el circuito, que es la zona de reconvergencia del circuito de la figura 3-19. Y en la figura aparecen tres de las 24 posibilidades de ordenación.

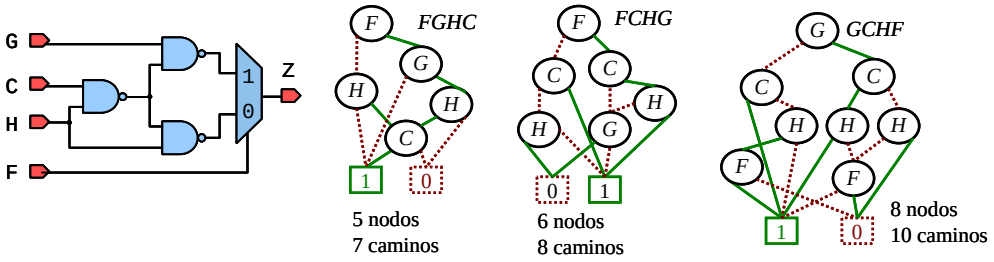


Figura 3-82: Tamaño de varios BDD del circuito según el orden de las variables

El orden RTL pondría la señal de la condición (*F*) en primer lugar, y las señales reconvergentes (*H* y *C*) en último lugar. Por tanto, en medio, quedaría la señal *G*. Así pues, resultan dos órdenes posibles según como se dispongan las señales reconvergentes: *FGHC* y *FGCH*.

Como se puede apreciar en la tabla 3-10, la distribución de los tamaños de los BDD es más compacta que en los dos ejemplos anteriores. Las dos ordenaciones proporcionadas por el RTL se sitúan en los cuatro órdenes con menor tamaño.

Orden	Nodos			Caminos		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
FGHC *- HCFG*	5	25	16	7	49	24
FGCH *- HCGF*	6	28	17	7	49	22
CHFG - GFHC	6	30	20	10	100	44
CFHG-FCHG-FHCG-FHGC GHFC-HFCG-HFGC-HGFC	6	34	22	8	64	31
CHGF-GFCH	7	33	21	10	100	40
CFGH-FCGH-GHCF-HGCF	7	37	23	8	64	31
CGFH-CGHF-GCFH-GCHF	8	45	28	10	100	44
Media	6,5	34,7	22	8,5	73.5	33,7
Mediana	6	34	22	8	64	31

Tabla 3-10: Tamaño de los BDD del ejemplo de la figura 3-82 según el orden de sus variables

3.6.1.5 Ejemplo 5

Este ejemplo considera un circuito que tiene una señal reconvergente que afecta a la alternativa de un multiplexor y a la vez afecta directamente a la salida. Es un circuito muy parecido al del ejemplo 3, con la diferencia de la reconvergencia de la señal *D*, que está unida mediante una puerta OR a la salida del multiplexor. El circuito, junto con los dos BDD más pequeños que se pueden formar se muestran en la figura 3-83, las cuatro permutaciones que forman estos BDD (*DAFME*, *DAMFE*, *AFMED*, *AMFED*) son las únicas de las 120 cuyos BDD tienen 6 nodos. Siendo precisamente estas cuatro permutaciones las que se obtienen siguiendo el orden RTL propuesto.

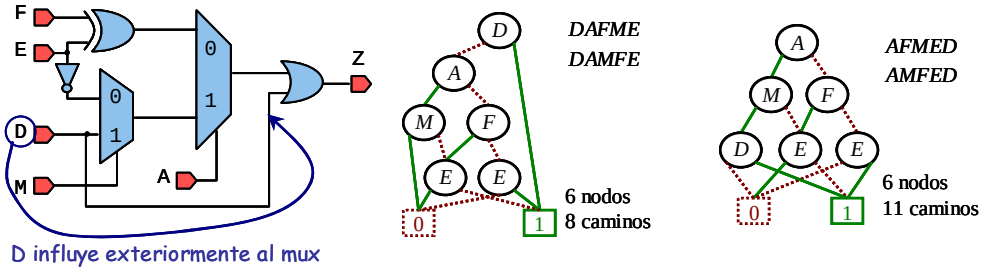


Figura 3-83: Los BDD más pequeños para circuito

Los órdenes *DAFME* y *DAMFE* se obtienen siguiendo los recorridos mostrados en la figura 3-84. Tomando la señal *Z* se llega a la puerta OR, a partir de ella, se pueden tomar cualquiera de sus dos entradas. En la figura se toma la señal *D*, y debido a que es mucho menos compleja que la otra entrada, la señal reconvergente (*D*) no se deja para el final, sino que se añade al orden desde que aparece (regla 9). Posteriormente se llega a la sentencia condicional de mayor jerarquía, llegados a este punto los dos recorridos posibles son similares a los de la figura 3-81, con la diferencia de que al llegar a la señal *D* no se añade por estar ya apuntada.

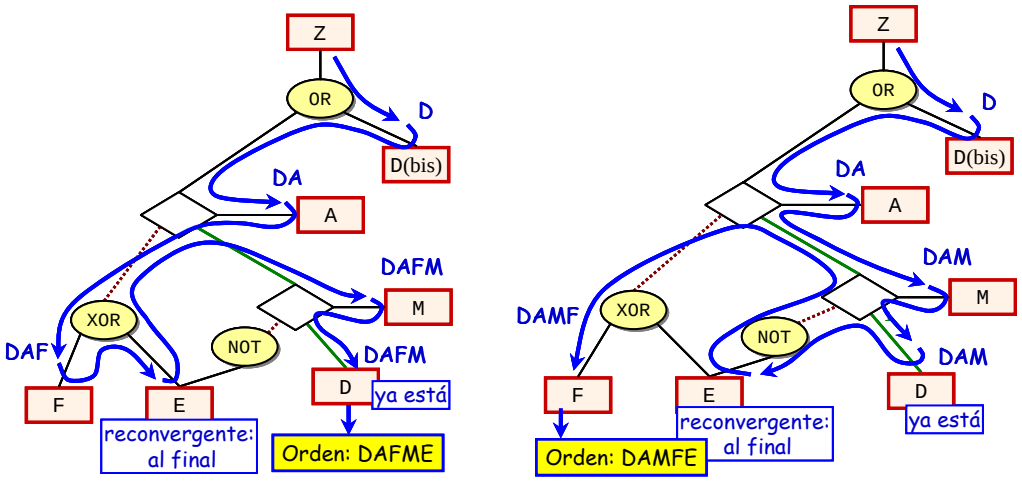


Figura 3-84: Recorrido del EHM tomando primero la entrada D de la puerta OR para la obtención del orden de las variables del BDD del ejemplo 5

Los otros dos órdenes (*AFMED* y *AMFED*) se obtienen tomando primero la entrada de la puerta OR que lleva al multiplexor (rombo en la figura 3-85). Como esta entrada es mucho más compleja que la otra (regla 7) la señal reconvergente de este camino se deja para el final de su área de reconvergencia.

Una vez llegados al multiplexor se añade la señal *A* al orden (regla 1). Posteriormente se puede escoger indistintamente el camino de la rama cero (recorrido de la izquierda de la figura 3-85) o el de la rama uno (derecha). En ambos recorridos, al ir de vuelta por la sentencia condicional de mayor jerarquía, se termina la región de reconvergencia de la señal *E*, por lo que se añade al orden. La señal *D* se añade después de la *E* por ser su región de reconvergencia mayor.

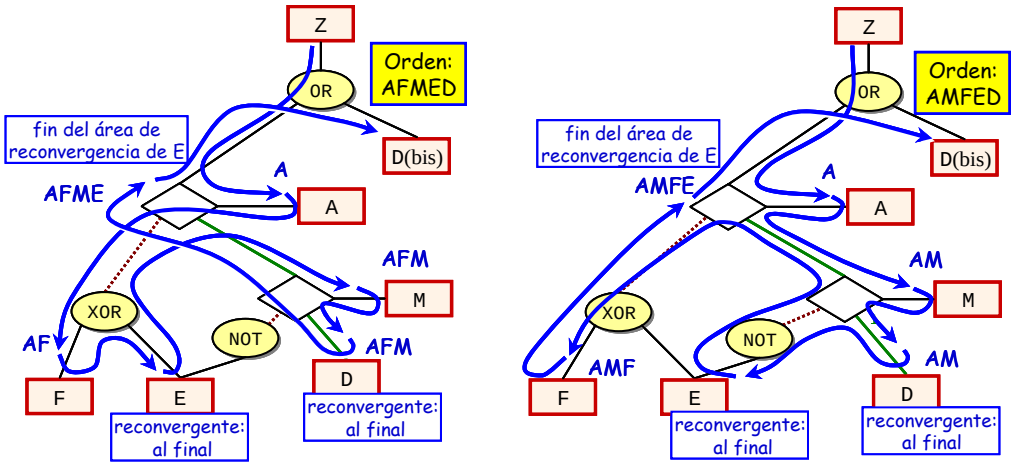


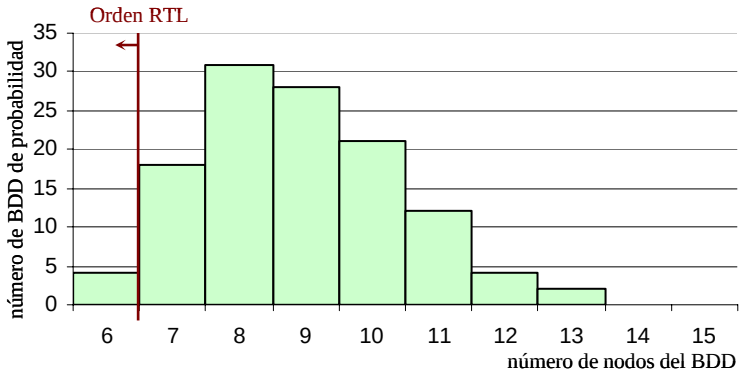
Figura 3-85: Recorrido del EHM tomando primero la entrada del multiplexor de la puerta OR para la obtención del orden de las variables del BDD del ejemplo 5

En la tabla 3-11 se han mostrado algunas de las 120 ordenaciones posibles. Las dos primeras filas corresponden a los únicos casos donde se forman BDD con 6 nodos. Estos casos corresponden con los el ordenamiento RTL. Obsérvese cómo los *a*BDD de la primera fila son menores que los de la segunda, pero no así los TFBDD, sin embargo el número de caminos de ambos BDD es menor para los de la primera fila. Al final de la tabla se muestra la media, mediana, el primer cuartil y el primer decil de toda la distribución. Los órdenes propuestos son menores que el primer decil en cuanto a número de nodos, en cuanto a número de caminos, los órdenes *AFMED* y *AMFED* corresponden con el primer cuartil.

Orden	Nodos			Caminos		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
DAFME - DAMFE	6	29	17	8	64	25
AFMED - AMFED	6	28	18	11	121	44
DAMEF	7	37	20	8	64	25
ADFME - ADMFE	7	34	21	9	81	34
AFMDE - AMFDE	8	41	26	11	121	54
AEMDF - EAMDF	9	53	32	11	121	58
EFMDA - FEMDA	10	61	38	14	196	90
FMADE - MFADE	11	63	43	19	361	153
FMDAE - MFDAE	12	84	56	16	256	109
FMDEA - MFDEA	13	93	61	16	256	111
[...]						
Media	8,88	51,3	32,95	12,82	173,8	71,23
Mediana	9	48,5	32	12,5	156,5	62
1er cuartil	8	42	26	11	121	48
1er decil	7	37	22	9	81	38

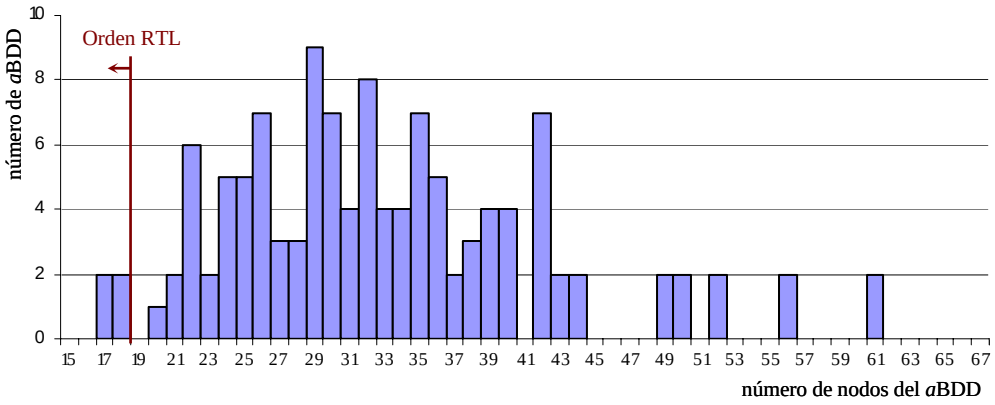
Tabla 3-11: Tamaño de algunos los BDD del ejemplo 5 según el orden de sus variables

Para dar una idea de la distribución de los tamaños de los BDD, en la gráfica 3-3 se ha representado en un diagrama de barras la cantidad de BDD con un número de nodos determinado. Los cuatro BDD obtenidos por la ordenación RTL son los que se encuentran en la primera barra. Comparándolo con la gráfica 3-1 se puede observar que para este circuito, que tiene más reconvergencias que el del ejemplo 3, la distribución es más compacta, pues no hay ningún BDD de 14 ni 15 nodos.



Gráfica 3-3: Distribución de los BDD de probabilidad del ejemplo 5 según su número de nodos

La distribución de los *a*BDD se muestra en la gráfica 3-4, aquí los cuatro órdenes RTL son las dos barras que están más a la izquierda. De manera similar a la distribución de los BDD de este ejemplo, este histograma es más compacto que el del ejemplo 3 (gráfica 3-3).



Gráfica 3-4: Distribución de los aBDDdel ejemplo 5 según su número de nodos

El tamaño de los BDD obtenidos por el orden propuesto (17 ó 18 nodos) es significativamente más bajo que los obtenidos con el peor caso para los TFBDD (93 nodos) o los *a*BDD (61 nodos). Y también lo es frente a los valores de la mediana: 48-49 nodos con TFBDD y 32 nodos con *a*BDD.

3.6.1.6 Ejemplo 6

Para ver si el orden RTL sigue siendo favorable para circuitos mayores, este ejemplo va a ampliar en una señal de entrada el ejemplo 5. Ahora, el circuito actual (figura 3-86) tiene 6 entradas, lo que implica que hay 720 órdenes posibles.

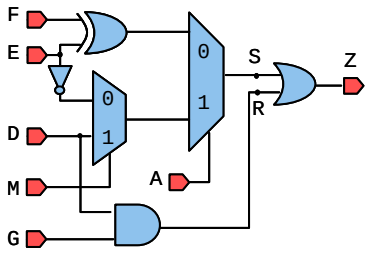


Figura 3-86: Circuito del ejemplo 6

Siguiendo el orden RTL se pueden obtener 8 disposiciones distintas de las señales. Cuatro de ellas salen de tomar primero la entrada de la OR que viene de la puerta AND (señal *R*). Estos recorridos sobre el EHM se muestran en la figura 3-87. Siendo su explicación similar al del ejemplo 5, con la diferencia que ahora de esta rama se obtienen tienen dos señales (*D* y *G*) y por

tanto, salen dos posibles órdenes por cada camino, que son *GDAMFE*, *DGAMFE*, *GDAFME* y *DGAFME*. De la misma manera, debido a que la señal *R* es mucho más sencilla que la *S*, no se deja la señal reconvergente *D* para el final (regla 9) sino que se añade según aparece.

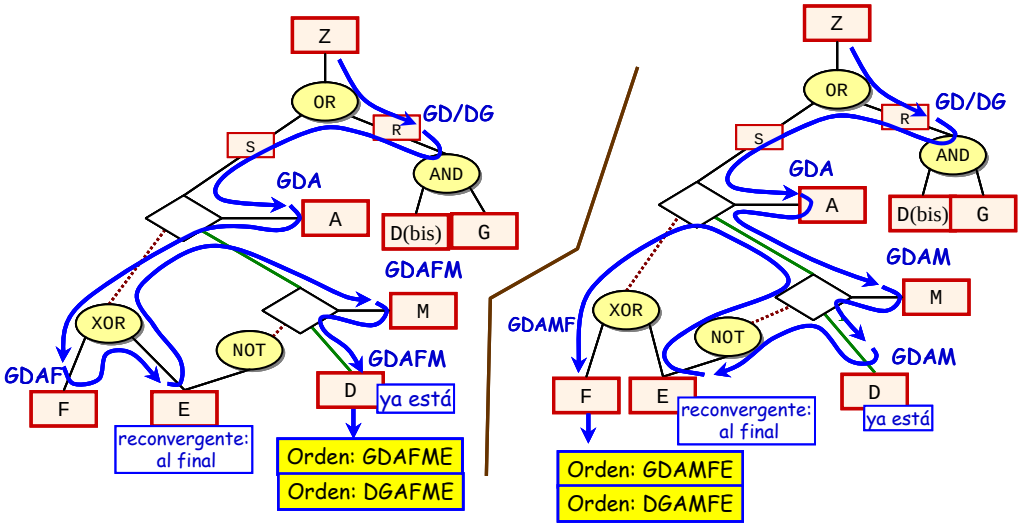


Figura 3-87: Recorridos del EHM para el ejemplo 6 tomando primero la señal R

La otra alternativa es tomar primero el camino que va hacia la sentencia condicional (la señal *S*), para por último tomar el camino que va hacia la señal *R*. Eligiendo estos caminos se obtienen los órdenes *AMFEGD*, *AMFEDG*, *AFMEGD* y *AFMEDG*, ya que se ha dejado la señal *E* al final de su zona de reconvergencia y la señal *D* al final del todo, pues su reconvergencia cubre todo el circuito. Los órdenes *AMFEDG* y *AFMEDG* se pueden considerar válidos por la regla 8 ya que ponen la señal reconvergente *D* al terminar el recorrido por la señal *S*. El recorrido de este camino no se ha dibujado ya que es similar al de la figura 3-85 del ejemplo 5.

En la tabla 3-12 se han puesto los órdenes de variables que dan los tamaños de BDD más pequeños y los dos mayores. En negrita se han señalado aquellos obtenidos por el orden RTL, y al final se incluyen la media, la mediana, el primer cuartil y el primer decil para los 720 casos.

Orden	Nodos			Caminos		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMEGD - AMFEGD	7	37	23	14	196	74
FAMEGD	8	42	27	20	400	138
MAFEGD	8	42	27	22	484	158
AFMEDG - AMFEDG	8	43	27	14	196	80
AMEFGD	8	51	32	14	196	80
FAMEDG	9	48	31	20	400	152
MAFEDG	9	48	31	22	484	168
DGAMFE	9	52	32	15	225	76
DGAFME	9	55	34	15	225	76
AFEMGD	9	54	35	16	256	102
AEFMGD - EAFMGD	9	56	36	16	256	102
MAEFGD	9	56	36	22	484	170
AMEFDG	9	57	36	14	196	86
AEMFGD - EAMFGD	9	60	38	16	256	102
DGAMEF	10	60	35	15	225	76
GDAMFE	10	57	36	22	484	144
GDAFME	10	60	38	22	484	144
[...]						
FGMEDA - FMGEDA - GFMEDA - GMFEDA - MFGEDA - MGFEDA	21	245	152	37	1369	534
EGFDAM - FGEDAM - GEFDAM - GFEDAM	22	206	131	31	961	423
Media	14,9	124,8	80,45	24,48	636,2	249,1
Mediana	15	121	76	24	576	229
1er cuartil	13	89	57	19	361	148
1er decil	11	69	45	17	289	115,9

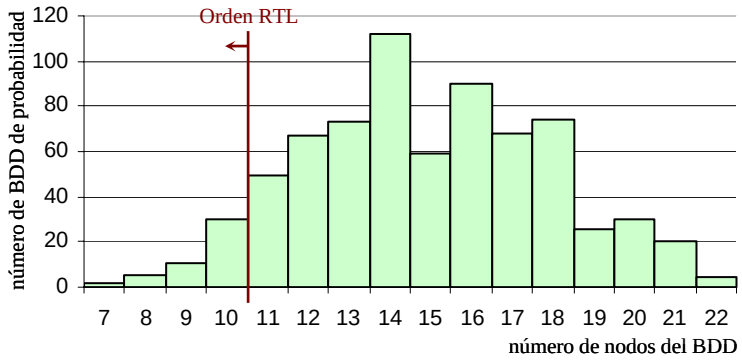
Tabla 3-12: Tamaño de los BDD más pequeños y mayores para el ejemplo 6

Los órdenes obtenidos por el RTL dan BDD pequeños, más para el caso que toma primero la señal *S* (sentencia *if*) en vez de la señal *R* (puerta AND). Generalmente es así, ya que suele dar órdenes menores si se escoge antes la señal que se le asigna una sentencia más compleja o profunda. Para este caso, los órdenes *AFMEGD* y *AMFEGD* del RTL son los óptimos (obsérvese que tienen la señal reconvergente *D* al final). Compárese la diferencia entre éstos: 7 nodos para el BDD y 23 nodos para el *a*BDD, frente a los 22 nodos para el BDD y 152 nodos del peor caso de los *a*BDD (penúltima fila), resultando más del triple para los BDD y más de 6 veces para los *a*BDD. Habiendo todavía más diferencia si se comparan los nodos del *a*BDD del caso óptimo (23) frente a los nodos del TFBDD del por caso (245).

Incluso el peor caso del orden obtenido por RTL (*GDAFME*) es todavía bastante ventajoso incluso comparado con la mediana: 10 nodos BDD frente a 15; y 38 nodos *a*BDD frente a 76 (justo la mitad).

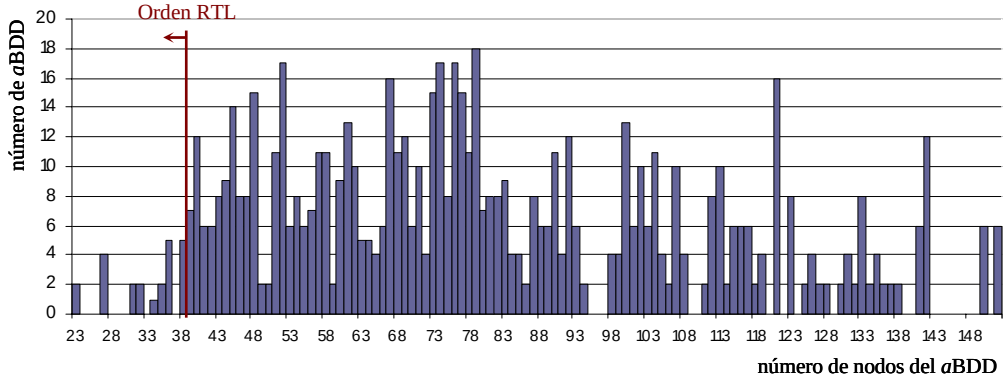
Obsérvese también que considerando el número de nodos, todos los órdenes RTL están por debajo del primer decil. Referido a número de caminos, todos están por debajo del primer decil a excepción de *GDAMFE* y *GDAFME* que están entre la mediana y el primer cuartil para los BDD de probabilidad y los TFBDD, y entre el primer cuartil y el primer decil para los *a*BDD.

La distribución de los BDD de probabilidad se ha dibujado en la gráfica 3-5, obsérvese que los órdenes RTL se encuentran en las cuatro primeras barras.



Gráfica 3-5: Distribución de los BDD de probabilidad del ejemplo 6 según su número de nodos

También se ha dibujado el histograma de los tamaños de los *a*BDD. Como es habitual, para los *a*BDD la gráfica está más distribuida que para los BDD, y por tanto la diferencia entre los extremos es más notable. Obsérvese dónde están los órdenes RTL (≤ 38) respecto a la distribución total.



Gráfica 3-6: Distribución de los aBDDdel ejemplo 6 según su número de nodos

3.6.1.7 Ejemplo 7

Para demostrar la importancia de la estructura del circuito a la hora de escoger el orden, en este ejemplo se van a poner varios circuitos con la misma estructura que la del ejemplo 6 pero con variaciones en las puertas lógicas. Por tanto, todos tendrán la misma estructura del EHM y tendrán idénticos recorridos del EHM para obtener los órdenes. En consecuencia, todos darán el mismo orden RTL. Estos cuatro circuitos se han dibujado en la figura 3-88.

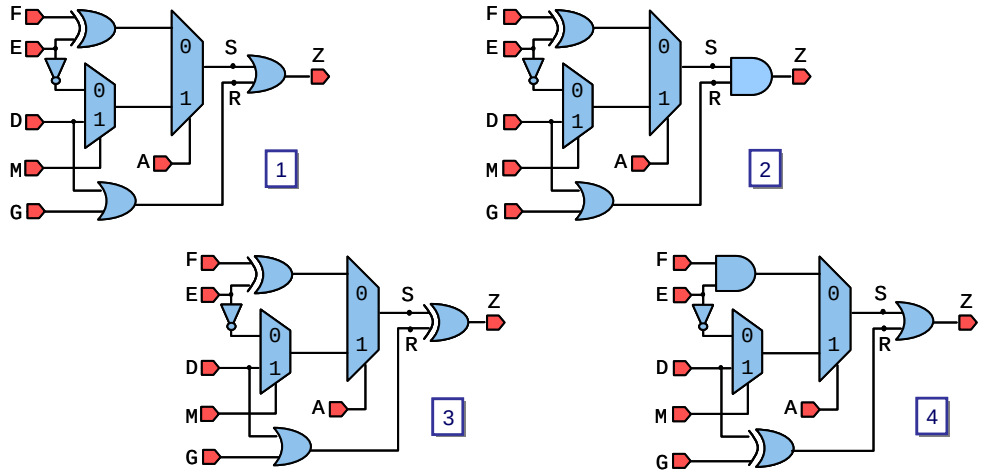


Figura 3-88: Los cuatro circuitos del ejemplo 7

Así, las ocho ordenaciones RTL se han puesto en la tabla 3-13 indicando el número de nodos de los BDD de probabilidad, los TFBDD y los *a*BDD para cada uno de los cuatro ejemplos. Después de estas ocho filas aparecen cuatro filas seguidas que indican casos máximos y mínimos. Así, la que indica "*Mín. BDD*" ofrece un caso con el mínimo número de nodos para el BDD; la siguiente es igual pero para *a*BDD. Las otras dos dan un caso con el número máximo de nodos para BDD y para *a*BDD. Luego vienen cuatro filas que dan el valor medio, la mediana, el primer cuartil y el primer decil de los 720 órdenes. Las dos últimas filas ofrecen la reducción en tanto por ciento del número de nodos del peor caso (% red. max.) y de la mediana (% red. med.) respecto del caso más favorable de orden RTL.

Orden	Nodos circuito 7.1			Nodos circuito 7.2			Nodos circuito 7.3			Nodos circuito 7.4		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMEGD	7	32	21	7	37	23	9	44	28	9	47	30
AMFEGD	7	32	21	7	37	23	9	44	28	9	47	30
AFMEDG	7	32	21	8	43	27	9	44	28	9	47	30
AMFEDG	7	32	21	8	43	27	9	44	28	9	47	30
DGAMFE	7	33	20	9	52	32	10	57	36	10	57	37
DGAFME	7	33	20	9	55	34	10	60	38	10	60	39
GDAMFE	7	33	20	10	57	36	10	57	36	10	57	37
GDAFME	7	33	20	10	60	38	10	60	38	10	60	39
[...]												
Mín. BDD	7	32	21	7	37	23	9	44	28	9	47	30
Mín. aBDD	7	33	20	7	37	23	9	44	28	9	47	30
Max. BDD	17	133	91	22	206	131	23	249	158	22	244	159
Máx. aBDD	17	133	91	21	245	152	23	249	158	22	244	159
Media	10,9	68,2	45,3	14,9	124,8	80,2	16,6	141,0	90,6	15,2	126,0	82,9
Mediana	11	65	43	15	121	76	17	137	86,5	15	212	78
1er cuartil	9,75	54	36	13	89	57	15	103,5	65	13	101	67
1er decil	9	47	31	11	69	45	13	80	52	12	78,9	52
% red. max.	58,8	75,9	78,0	68,2	84,9	84,9	60,9	82,3	82,3	59,1	80,7	81,1
% red. med.	36,4	50,8	53,5	53,3	69,4	69,7	47,1	67,9	67,6	40,0	77,8	61,5

Tabla 3-13: Tamaño de los BDD del orden RT para los cuatro circuitos; tamaños mínimos, máximos, medios y mediana considerando todos los órdenes; reducción de nodos de los tamaños máximos y de la mediana respecto al tamaño mínimo del orden RTL

Nótese cómo tomando primero el camino de la señal *S* dan los mejores órdenes. Así pues, como se ha comentado, **tomar antes el camino de la señal más compleja suele resultar en mejores órdenes**.

Analizando la tabla se puede observar que en todos los casos hay más de un orden RTL que se corresponde con el orden óptimo. Por otro lado, el peor de los órdenes RTL es siempre bastante más favorable que el de la mediana, de hecho, **todos los casos están por debajo del primer decil**. La reducción de nodos para el mejor caso RTL respecto a la mediana es superior al 50% en los BDD de actividad, y superior al 36% en caso de los BDD de probabilidad.

3.6.1.8 Ejemplo 8

Un ejemplo más complejo se muestra en la figura 3-89. Éste tiene 7 entradas, por tanto existen 5040 posibilidades de ordenación.

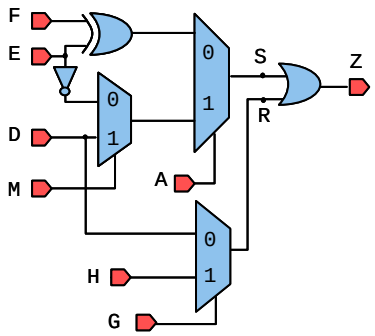


Figura 3-89: Circuito del ejemplo 8

Del recorrido RTL surgen 8 posibles órdenes que se han incluido en la tabla 3-14, que son *AFMEDGH*, *AMFEDGH*, *AFMEGHD*, *AMFEGHD*, *GHDAMFE*, *GHDAFME*, *GDHAMFE* y *GDHAFME*.

Los órdenes *AFMEDGH*, *AMFEDGH*, *AFMEGHD* y *AMFEGHD*, se obtienen por la regla 7, tomando antes la señal *S* que es más compleja que la *R*. Al terminar el recorrido por la señal *S*, por la regla 8 se puede añadir la señal *D* (*AFMEDGH*, *AMFEDGH*) o bien ponerla al final de toda su región de reconvergencia (*AFMEGHD* y *AMFEGHD*).

Los órdenes *GHDAMFE*, *GHDAFME*, *GDHAMFE* y *GDHAFME* se obtienen por la regla 9, tomando antes la señal más simple *R*, y no dejando para el final la reconvergente *D*, sino añadiéndola según aparezca. Nótese que estos órdenes son menos favorables.

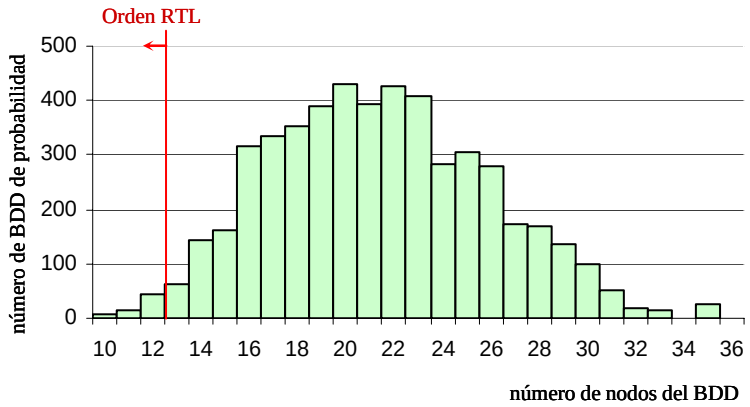
En la tabla también se han incluido las peores ordenaciones posibles (*HMFDEGA*, *HFMDEGA* y *MHFDEGA*). También se muestran se muestran la media, mediana, primer cuartil, primer decil y el quinto percentil. Todos los órdenes RTL están debajo del percentil quinto, excepto los de la primera fila (*AFMEDGD*, *AMFEDGH*) respecto al número de caminos del BDD y el TFBDD. Curiosamente éstos son los óptimos en número de nodos, aún así se encuentran debajo del primer decil.

En las dos últimas filas de la tabla se muestra el porcentaje de la reducción de nodos y caminos del peor valor (% red. max.) respecto al de la primera fila; y del valor de la mediana (% red. mediana) respecto a la primera fila. La primera fila tiene un orden óptimo para el número de nodos, pero no así para el número de caminos, saliendo por tanto unos porcentajes menores. Nótese cómo, comparando con el peor caso, el número de nodos en los BDD de actividad es más de diez veces mayor, y más de tres veces mayor para el BDD de probabilidad. Comparando con la mediana, el número de nodos de los BDD de actividad de la primera fila es cuatro veces menor, y dos veces menor para el BDD de probabilidad.

Orden	Nodos			Caminos		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMEDGH - AMFEDGH	10	55	36	25	625	219
AFMEGHD - AMFEGHD	10	59	38	20	400	152
GHDAMFE	11	64	41	23	529	167
GHDAFME	11	67	43	23	529	167
GDHAMFE	12	73	47	24	576	191
GDHAFME	12	76	49	24	576	191
[...]						
HMFDEGA - HFMDEGA - MHFDEGA	35	587	374	52	2704	1176
Media	21,4	235,4	153,1	37,9	1523,8	617,3
Mediana	21	221	145,5	38	1444	578
1er cuartil	18	162	106	30	900	395
1er decil	16	124	80,9	26	676	293
percentil 5	14	104	67	24	576	240
% red. max.	71,4	90,6	90,4	51,9	76,9	81,4
% red. mediana	52,4	75,1	75,3	34,2	56,7	62,1

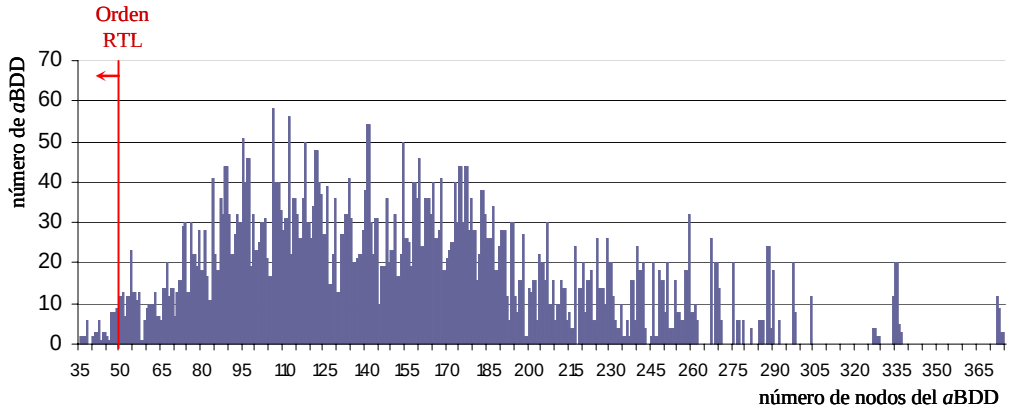
Tabla 3-14: Tamaño de los BDD obtenidos del orden RTL y los órdenes que dan los mayores BDD para el ejemplo 8; y algunas medidas de posición para la distribución

Para tener una idea de la distribución y de donde se encuentran los órdenes propuestos, en la gráfica 3-7 se muestra el histograma del tamaño de los BDD. Nótese dónde se encuentran los BDD propuestos, que tienen un número de nodos igual o menor que 12 (tercera barra).



Gráfica 3-7: Distribución de los BDD de probabilidad del ejemplo 8 según su número de nodos

La distribución del número de nodos de los *a*BDD se ha dibujado en la gráfica 3-8. Obsérvese dónde están los *a*BDD del orden RTL en la gráfica, teniendo todos un número de nodos menor que 50.



Gráfica 3-8: Distribución de los aBDDdel ejemplo 8 según su número de nodos

3.6.2 Conclusiones sobre el orden RTL

A la vista de los ejemplos mostrados, puede considerarse que **el orden proporcionado por el análisis RTL es adecuado**. Se han realizado más pruebas con circuitos de tamaño parecido con los que se han obtenido resultados similares, estas pruebas no se han incluido pues se ha considerado no aportan ninguna novedad significativa a los resultados mostrados.

Como se ha podido ver, **la mayoría de los órdenes propuestos están debajo del primer decil** y en muchos casos, **algunos estos órdenes resultan ser el óptimo** de todas las posibilidades.

A partir de las variaciones realizadas de un mismo circuito con distintas puertas lógicas, podría deducirse que la elección de un buen orden depende mucho de la estructura. Aunque también la función lógica afecta, ya que como se ha visto, cuanto más complicada sea la función lógica de una señal más adecuado suele ser recorrer la estructura de esa señal antes que el camino de otras señales más sencillas (ejemplos 6 y 7).

Las reglas proporcionadas son abiertas, y dan varios órdenes. En general se puede decir que cualquiera de ellos es válido, pero si se pretende escoger un único orden, la regla sería coger primero la alternativa o señal más compleja (regla 6), y dejar la señal reconvergente para el final (regla 5). De todos modos, la propuesta de varios órdenes puede considerarse una ventaja en el sentido de que, en vez de arriesgar por un único orden, permite la elección del orden óptimo dentro de un conjunto reducido. Por ejemplo en el circuito 8, de los 5040 órdenes posibles se ofrecen 8 posibilidades que el usuario puede construir y comparar para escoger el más conveniente.

Por último, conforme el número de entradas aumenta, la diferencia entre los extremos de la distribución se hace mayor, por lo tanto se hace más crítica la elección de un orden adecuado. Según los ejemplos que se han estudiado, el orden RTL se mantiene en las posiciones más bajas del histograma, reduciendo incluso el percentil donde se encuentra para tamaños de circuito mayores. Este método, sin ser definitivo y sin tener una base teórica que lo sustente, parece que **ofrece un sistema válido e inmediato para la elección del orden de las variables de los BDD**.

3.7 Propagación de probabilidades y actividades

Hasta ahora se ha visto cómo a partir de un circuito se obtiene su modelo extendido del hardware (EHM, §3.3.2). Este modelo se recorre desde las señales de menor profundidad combinacional a las de mayor (§3.3.3) realizando el análisis de regiones de reconvergencia y disjuntas (§3.4) para lograr particiones más pequeñas. A partir de estas regiones y a través de la estructura proporcionada por el EHM se construyen los BDD de probabilidad y actividad (§3.5). Para estos BDD de actividad se ha propuesto una forma compacta de representarlos (§3.5) y un método de ordenación basado en la estructura RTL (§3.6).

Por tanto, en este punto se dispone del BDD de probabilidad y actividad de la señal. Debido a que el análisis se realiza empezando por las señales de menor profundidad combinacional, se tienen los valores numéricos de las todas dependencias de la señal. Así pues, ya se puede realizar el cálculo numérico de la probabilidad y actividad de la señal. En el anexo AI.2 se resume el algoritmo empleado para calcular los valores numéricos de la probabilidad y actividad a partir de los BDD.

Una vez obtenido los resultados numéricos de la probabilidad y actividad, éstos se apuntan en el EHM. De esta manera estos valores están accesibles para los cálculos del resto de señales de mayor profundidad combinacional.

Antes de realizar la propagación de la probabilidad y actividad a lo largo del circuito es necesario que el usuario provea de los valores numéricos de probabilidad y actividad de los puertos de entrada del circuito.

La propagación de probabilidades y su anotación en el modelo del hardware (EHM) resulta en un mapa de actividad como el mostrado en la figura 3-90 para un circuito de ejemplo.

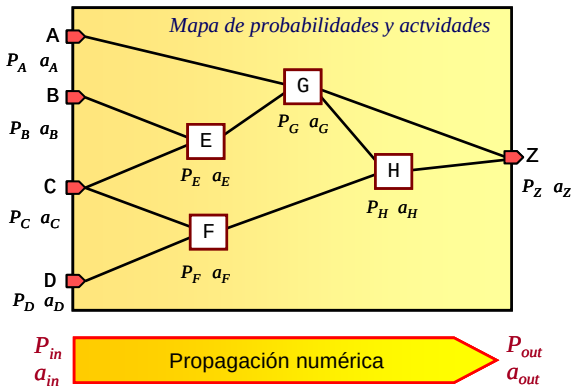


Figura 3-90: Ejemplo de un mapa actividad resultante de la propagación

Una vez que se ha realizado el análisis y se ha obtenido el mapa de actividad, para conocer las probabilidades del circuito con otras condiciones de entrada, basta con realizar esta última propagación de probabilidades y actividades. Es decir, todo el procesamiento previo realizado en los apartados anteriores se mantiene en el modelo extendido del hardware, y sólo se tienen que propagar los valores numéricos bajo la misma estructura y BDD. Por tanto, este último procesamiento es muy **rápido**, lo que proporciona un **medio eficaz para conocer la actividad del circuito en diferentes situaciones**. **Esto es una clara ventaja frente el método simulativo** (§2.1.1), que requiere de una nueva simulación completa cada vez que se cambian las condiciones de entrada.

3.8 Conclusiones

En este capítulo se ha detallado la propuesta de estimación de actividad de esta tesis doctoral. Esta propuesta consiste en subir el nivel de abstracción del cálculo probabilístico de la actividad de conmutación, pasando del nivel lógico o de puertas al nivel de transferencia de registros (RTL). El método es la primera propuesta a la resolución del problema en este nivel y por tanto no aspira a ser completo, sin embargo, las ventajas que se derivan del análisis probabilístico desarrollado para RTL alientan a profundizar en el método.

Subir de nivel de abstracción tiene muchas implicaciones en el método de estimación. Por tanto, la propuesta de esta tesis ha tenido que enfrentarse a nuevas dificultades, aunque también, se ha visto favorecida por las ventajas que supone el análisis en un nivel más alto. Y así, la propuesta general de esta tesis se compone de varias propuestas concretas que salvan las dificultades y se aprovechan de la nueva perspectiva. Todas estas propuestas individuales contribuyen al objetivo principal de la tesis que es **subir el nivel de abstracción de la estimación de la actividad de conmutación** desde el nivel de puertas al nivel RT.

A causa de la diversidad de propuestas individuales, para favorecer una lectura fluida y proporcionar una información menos densa, algunas secciones no se han incluido en este capítulo, sino que se han añadido al anexo II. Estas secciones adicionales, si bien complementan y son útiles para profundizar en las aportaciones, no son imprescindibles para entender las propuestas que en esta tesis se hacen.

Seguidamente se resumirán las propuestas y ventajas del nuevo método.

La **propuesta principal de esta tesis**, que es subir el nivel de abstracción de la estimación de la actividad de conmutación, **permite realizar el análisis con anterioridad** cuando se emplean metodologías descendentes (*top-down*). Los resultados de la estimación **ayudan a la toma correcta de decisiones** de diseño. Como ahora estas estimaciones están disponibles en una

etapa menos avanzada del diseño, el método contribuye a **disminuir el número de iteraciones en el flujo del diseño**.

Por otro lado, subir el nivel de abstracción a RTL también proporciona ventajas prácticas en el análisis a realizar. El nivel RT proporciona:

- Una **mayor información de alto nivel** que no está disponible en el nivel de puertas, esta información la proporcionan los operadores lógicos y matemáticos, la definición de estructuras como los multiplexores, y el agrupamiento de señales en vectores.
- Una **mayor simplicidad del análisis** ya que ahora el diseño no tiene una información tan exhaustiva como la que tenía en el nivel de puertas.

Sin embargo, esta simplificación del diseño tiene una contrapartida, ya que para el análisis es necesaria parte de la información que no está en el nivel RT. Un ejemplo de esto es la falta de una estructura de dependencias e influencias de las señales como la que hay en el nivel de puertas (*netlist*). Y por esto, en esta tesis **se propone la creación de un modelo del hardware** (EHM) que proporcione una estructura sobre la que se realice el análisis probabilístico (§3.3). Las ventajas de este modelo del hardware son:

- Proporcionar la **estructura** de dependencias e influencias similar a la *netlist* del nivel de puertas.
- **Conservar la información de alto nivel** para utilizarla en el análisis probabilístico.
- **Mantener la sencillez**, evitando los inconvenientes de un modelo tan detallado como el del nivel de puertas.

Para circuitos grandes es necesario realizar particiones para limitar la necesidad de recursos computacionales y de memoria. Sin embargo, las particiones deben considerar las dependencias estructurales para no cometer errores en el cálculo. La partición en regiones reconvergentes proporciona valores exactos de probabilidad y actividad. En este ámbito, realizar el análisis en RTL frente a puertas tiene la ventaja de:

- **Mantener la reconvergencia**, ya que en ocasiones, al sintetizar un circuito en puertas, su reconvergencia aumenta y las regiones resultantes son mayores o no es posible la partición.
- El particionado es **más sencillo** que en puertas debido a que el modelo en RTL no es tan detallado.

No obstante, en muchos casos, las regiones resultantes siguen siendo excesivamente grandes o incluso hay circuitos con los que no se logra hacer partición. Por ello, en esta tesis **se propone la partición en regiones disjuntas desde el nivel RT** (§3.4) por su **menor tamaño** respecto a las regiones reconvergentes. Si bien este tipo de particiones ya había sido propuesto para el nivel de puertas [2], solamente se hizo para la estimación de probabilidad. Además, al contrario de lo que ocurre en el nivel de puertas, la identificación de estas regiones en RTL es **sencilla** (§3.4.1). En consecuencia, en esta tesis **se propone la detección de las regiones disjuntas en RTL**.

El cálculo de la probabilidad mediante regiones disjuntas es exacto, sin embargo, para el cálculo de la actividad se realizan ciertas aproximaciones. En esta tesis **se ha evaluado la incidencia de estas aproximaciones en el valor de actividad resultante** (§3.4.3). Como resultado el estudio establece unas **condiciones bajo las que se puede realizar la partición con un error aceptable**.

En esta tesis se usan BDD para realizar los cálculos de probabilidad y actividad. Las ecuaciones y los BDD de actividad son significativamente más complejas que las de probabilidad. Por esto, en esta tesis **se propone una nueva forma de representar los BDD de actividad** (§3.5.2). Asimismo, se Incluye una formalización teórica de esta propuesta (§3.5.2.4). Con esta propuesta logra:

- **Reducir entre un 25% y un 50% el tamaño de los BDD de actividad**. Y con reducciones mayores en el número de caminos. Todo ello sin detrimento en la exactitud.
- La formalización teórica permite la creación de una función de actividad que se ha integrado en el paquete informático de los BDD (BuDDy [64]).

El orden de las variables de los BDD tiene mucha influencia en su tamaño. Los algoritmos de ordenación son complejos ya que para un BDD de *n* variables existen *n*! órdenes distintos. En consecuencia, en esta tesis **se propone ordenar las variables según la estructura RTL del circuito** (§3.6). En esta tesis se ha realizado un estudio empírico de los distintos órdenes de variables según la estructura RTL. Como resultado **se han propuesto unas reglas que permiten ordenar los BDD de manera sistemática y eficiente**. Los resultados de esta ordenación son:

- Al contrario de lo que ocurre en el nivel de puertas, esta manera de ordenar **resulta sencilla de llevar a cabo desde el nivel RT**. y por lo tanto no se necesita recurrir a complejos algoritmos de ordenación
- La creación de las reglas permiten la automatización del proceso de ordenación
- **Los resultados experimentales corroboran la validez del método**. De todas las posibilidades de ordenación, en la mayoría de los casos, los ordenamientos RTL se encuentran por debajo del primer decil.

Además en esta tesis **se propone hacer uso de la información relativa a los índices de los vectores que se dispone en el nivel RT** para lograr un mejor ordenamiento del BDD. El conocimiento de los índices de los vectores **permite ordenar las variables de los BDD de manera óptima según la operación** en la que estén implicados. Esta información **no se tiene en el nivel de puertas**. Este punto se ampliará en el capítulo de resultados (capítulo 4).

Por último, la propagación de los valores de probabilidad y actividad se realiza a través del modelo del hardware propuesto (EHM). Debido a que el método de estimación es estático, se requiere una significativa intensidad de cálculo para su creación, no obstante, una vez creado se pueden recalcular valores con gran rapidez y un bajo coste computacional.

Todas estas propuestas se han implementado en una herramienta CAD, que por un lado demuestra de manera práctica la viabilidad del método. Y por otro lado permite llevar a cabo multitud de pruebas con circuitos de tamaño considerable que sin una herramienta CAD sería inviable.

En conclusión, esta propuesta de estimación de la actividad de conmutación constituye la **base** para la estimación probabilística en el nivel RT. Esta propuesta se constituye de múltiples propuestas individuales que hacen posible la estimación en RTL y hacen uso de las ventajas que ofrece el nuevo nivel. En el capítulo siguiente se llevarán a cabo experimentos para confirmar la validez de las propuestas.

4. RESULTADOS EXPERIMENTALES

En este capítulo se demuestra experimentalmente la validez de la propuesta de estimación de esta tesis doctoral. La propuesta fundamental de esta tesis es subir la estimación probabilística de la actividad de conmutación del nivel de puertas a RTL. Las ventajas teóricas de esta propuesta se expusieron en el capítulo anterior. En este capítulo se comprueba que esas ventajas también se cumplen en la práctica.

En el capítulo anterior se ha visto que, por la magnitud de la propuesta, esta tesis aporta diversas soluciones que resuelven aspectos concretos del método y que conforman la propuesta principal. Estas propuestas particulares se resumen en:

1. Uso de un modelo extendido del hardware (EHM §3.3.2)
2. Partición del circuito en regiones disjuntas (§3.4)
3. Nueva representación de los BDD de actividad (*a*BDD §3.5.2)
4. Propuesta de ordenamiento RTL de los BDD (§3.6)

Para la demostración de la propuesta principal y de las propuestas particulares se ha analizado el método de estimación con varios circuitos. De estos circuitos se ha obtenido su modelo extendido del hardware (EHM) para su versión en puertas, en RTL y en RTL realizando partición por regiones disjuntas. Para cada uno de estos modelos se han obtenido los BDD de actividad propuestos (*a*BDD) y los TFBDD para su comparación. Una vez que se tienen el EHM y los BDD, se propagan los valores numéricos de probabilidad y actividad desde las entradas a las salidas. Por último, se comparan los resultados obtenidos por cada uno de los métodos.

Para demostrar las ventajas del análisis RTL y probar las cuatro propuestas particulares, se ha analizado:

a. La variación del tamaño de los BDD obtenidos con al análisis en RTL sin partición por regiones disjuntas frente al obtenido en puertas. Con esto se prueban las ventajas del análisis RTL frente a puertas y las ventajas de ordenamiento RTL (propuesta 4).
b. La variación del tamaño de los BDD obtenidos con el análisis en RTL con partición por regiones disjuntas frente al obtenido en puertas y en RTL sin partición. Con esto se prueban las ventajas de la partición por regiones disjuntas (propuesta 2). Esta partición sólo se puede llevar a cabo de manera sencilla en RTL, por tanto, en sí misma es una ventaja del análisis en RTL frente a puertas. Como esta partición no produce resultados exactos, también se analiza el error cometido.
c. La variación de tamaño de los BDD de actividad empleando *a*BDD frente a TFBDD (propuesta 3)

El uso del EHM (propuesta 1) es un paso imprescindible para el análisis en RTL, ya que proporciona una estructura necesaria para la propagación de probabilidades y actividades (§3.3). Por tanto, su validez queda demostrada como medio para poder realizar el estudio de la actividad. Además, sin el EHM no se podría identificar de manera sencilla las zonas disjuntas del circuito. En consecuencia, el EHM es también paso intermedio necesario para llevar a cabo la propuesta 2. Por último, el EHM tiene la ventaja de ser un análisis de una sola vez. Una vez construido el EHM para un circuito no es necesario volverlo a elaborar aunque cambien las condiciones de las entradas.

Así, si en estos circuitos de pruebas se cumple que los análisis son favorables para el nivel RT frente al nivel de puertas, se corroborarán las demostraciones teóricas del capítulo anterior. En consecuencia, se podrá afirmar que la propuesta de esta tesis es válida.

El capítulo se estructura de la siguiente manera, en la sección 4.1 se realiza una breve descripción de los circuitos utilizados en las pruebas. Luego, en la sección 4.2 se comparan los diferentes modelos probabilísticos resultantes de cada circuito. Posteriormente, en la sección 4.3 se analiza el error cuando se realiza partición por regiones disjuntas. En la sección 4.4 se estudian los resultados de cada una de las propuestas particulares de manera conjunta con

todos los circuitos, en vez de realizar un estudio por circuito como el de las secciones anteriores. En la sección 4.5 se hace un breve apunte sobre los tiempos de procesamiento. Por último, en el apartado 4.6 se elaboran las conclusiones de este capítulo. Además, en el anexo III se ha incluido información y análisis adicionales a los presentados en este capítulo. Éstos se han apartado para simplificar la información presentada, aunque permite al lector interesado profundizar en dichos análisis.

4.1 Descripción de los circuitos de pruebas

En esta sección se exponen cada uno de los circuitos que se usan para el análisis. Los circuitos están descritos en VHDL.

Los diseños han sido obtenidos de dos fuentes. Los tres primeros se han obtenido de una serie de ejemplos de circuitos RTL para síntesis propuestos por Z. Navabi [144]. En esta fuente se ofrecen los diseños tanto en RTL como en puertas.

La segunda serie de diseños se ha obtenido de la ALU del microcontrolador 8051 proporcionado por *Oregano Systems* [101]. Los diseños de este circuito son genéricos y permiten variar el ancho de bus de los datos. Los diseños en nivel de puertas se han obtenido mediante la síntesis con el *Design Compiler™* de *Synopsys®* [125]. En algunos casos, cambiando las opciones de síntesis, se han obtenido varias versiones de un circuito en el nivel de puertas. Esto se ha realizado para analizar las diferencias entre varias versiones de un mismo circuito en puertas.

La tabla 4-1 muestra el resumen de los circuitos analizados. Para tener un abanico más amplio, algunos de ellos incluyen variaciones en el ancho del bus de datos. Esto además, permite estudiar cómo afecta el tamaño del circuito sobre cada uno de los análisis realizados: puertas, RTL y RTL-disjunto.

nº	Nombre del circuito	Ancho de bus	Puertas totales	Puertas de 2 entradas	Inv.	Puertos entrada	Puertos salida	Fuente	Descripción
1	max	4	26	22	4	8	4	[144]	Devuelve la entrada mayor
		5	34	29	5	10	5		
		6	42	37	5	12	6		
		7	50	44	6	14	7		
		8	57	51	6	16	8		
		9	65	58	7	18	9		
		10	71	65	6	20	10		
		11	79	72	7	22	11		
		12	87	79	8	24	12		
		13	97	86	11	26	13		
		14	105	93	12	28	14		
		15	110	100	10	30	15		
		16	117	107	10	32	16		
		17	127	114	13	34	17		
		18	132	121	11	36	18		
		19	139	128	11	38	19		
		20	147	135	12	40	20		
2	comparador	4	30	25	5	11	3	[144]	Comparador en cascada
		5	36	30	6	13	3		
		6	42	35	7	15	3		
		7	51	42	9	17	3		
		8	57	48	9	19	3		
		9	63	53	10	21	3		
		10	72	59	13	23	3		
		11	79	66	13	25	3		
		12	83	70	13	27	3		
		13	95	76	19	29	3		
		14	96	81	15	31	3		
		15	105	89	16	33	3		
		16	109	92	17	35	3		
		17	117	99	18	37	3		
		18	125	106	19	39	3		
		19	132	111	21	41	3		
		20	140	118	22	43	3		
		21	146	123	23	45	3		
		22	150	127	23	47	3		
3	alu_peq	4	510	371	139	11	7	[144]	ALU de 4 bits
4	alu8051_core	8	219	201	18	22	10	[101]	Núcleo de la ALU del 8051
5	addsub	7	70	62	8	16	10	[101]	Sumador/restador
		8	72	63	9	18	11		
		9	87	75	12	20	13		
		10	101	87	14	22	14		
		11	108	94	14	24	15		
		12	117	100	17	26	16		
		13	125	107	18	28	18		
		14	139	119	20	30	19		
		15	146	126	20	32	20		
6	alu8051_simp	7	383	348	35	23	10	[101]	ALU del 8051 simplificada
		8	415	375	40	25	11		
		9	450	412	38	28	13		
		10	487	447	40	30	14		
		11	517	474	43	32	15		
		12	557	503	54	34	16		
7	alu8051	6	520	483	37	27	9	[101]	ALU del 8051
		7	571	532	39	30	10		
		8	629	580	49	33	11		
		9	685	636	49	37	13		
		10	759	689	70	40	14		

Tabla 4-1: Resumen de los circuitos analizados

En la tabla 4-1, el número de puertas ("*Puertas totales*") se refiere al circuito sintetizado, y es un valor orientativo ya que éste depende de las opciones de síntesis. La síntesis se ha realizado con puertas de dos entradas e inversores. Para distinguirlas, se han incluido dos columnas que indican qué tipo de puertas son: "*Puertas de 2 entradas*" se refiere al número total de puertas de dos entradas; y la columna "*Inv*" se refiere al número de inversores del circuito.

Debido a que en esta tesis se propone realizar el análisis en RTL y el resto de propuestas realiza el análisis en el nivel de puertas, los resultados de esta tesis no se han podido comparar directamente con otros trabajos. La mayoría del resto de trabajos (por ejemplo [28], [119], [80], [7]) utilizan los circuitos del ISCAS'85 [15] para las pruebas. Estos circuitos están descritos en el nivel de puertas, y por tanto, con ellos no se pueden comprobar las ventajas del análisis en RTL. Como el objetivo es demostrar las ventajas del análisis en RTL, se ha tenido que recurrir a otros circuitos.

De la tabla 4-1 se observa que todos los circuitos escogidos son aritméticos. Esto se debe a que los circuitos de control son eminentemente secuenciales y, normalmente, su parte combinacional no es muy grande. Esto ocurre en los circuitos del ISCAS'85, que también son combinacionales y la mayoría son aritméticos: de los 10 circuitos propuestos, hay 4 unidades aritmético-lógica (ALU), un multiplicador, un sumador/comparador, un controlador de interrupciones y tres circuitos detectores de error (dos de ellos son el mismo pero descrito con distinto tipo de puertas) [53].

Además, normalmente, la parte combinacional de los circuitos de control consta principalmente de multiplexores. Como se vio en el capítulo 3, la existencia de multiplexores beneficia al análisis en RTL por facilitar el ordenamiento RTL (§3.6) y permitir la partición en regiones disjuntas (§3.4).

Se debe señalar que no se han tomado circuitos de fuentes más diversas por el esfuerzo de adaptación que requiere el análisis. Esto se debe a que el programa informático que se ha creado para construir los modelos y realizar las estimaciones es una versión preliminar que sólo acepta un conjunto reducido del VHDL (§3.2). Esto implica que cada circuito necesita ser adaptado manualmente. Entre las operaciones que se realizan para la adaptación figuran sustituir los vectores por señales de un bit y el aplanado del circuito (quitar la jerarquía y poner el todo el circuito en un único módulo). Estas operaciones implican un esfuerzo considerable, especialmente en circuitos grandes. Es por esto por lo que no se han tomado circuitos de diversas fuentes. Para el caso de la ALU del 8051, se han ido aplanando diversas partes de un mismo circuito y se han ido uniendo. Como estas partes realizan funciones diferentes, se puede considerar que los resultados son independientes. Por otro lado, este análisis incremental permite investigar las implicaciones en el estudio probabilístico del aumento de la funcionalidad y el tamaño de un circuito.

A continuación se detallan los circuitos utilizados en las pruebas.

4.1.1 Circuito "max" (1)

El primer ejemplo se ha obtenido de Navabi [144]. Es un circuito sencillo que devuelve la mayor de dos entradas de cuatro bits. En la figura 4-1 se muestra el esquema y el código VHDL de dicho circuito. El modelo en puertas lógicas proporcionado tiene 28 puertas de dos entradas (AND y OR) y 11 inversores.

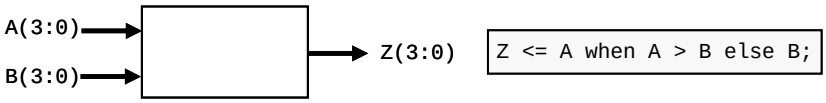


Figura 4-1: Esquema y código VHDL del circuito "max" (1)

4.1.2 Circuito "comparador" (2)

El segundo ejemplo también se ha obtenido de Navabi [144]. Éste es un comparador que se puede conectar en cascada. Indica cuál de las dos entradas de cuatro bits es mayor. En la figura 4-2 se muestra el esquema y el código VHDL de dicho circuito. El modelo en puertas lógicas proporcionado tiene 29 puertas de dos entradas (AND y OR) y 17 inversores.


```
Z <= "100" when A > B else
     "001" when A < B else
     C;
```

Figura 4-2: Esquema y código VHDL del circuito "comparador" (2)

4.1.3 Circuito "alu_peq" (3)

El tercer circuito (Navabi [144]) es una ALU simple que cuenta con ocho operaciones para las entradas *A* y *B*, que son de cuatro bits. Las operaciones están definidas por la entrada *Code*, el resultado y el estado de la operación los ofrecen las salidas *Z* y *Flags* respectivamente. El modelo proporcionado en puertas lógicas tiene 371 puertas de dos entradas (AND y OR) y 139 inversores. En parte, el alto número de puertas lógicas se debe a que no se han utilizado puertas XOR ni XNOR.

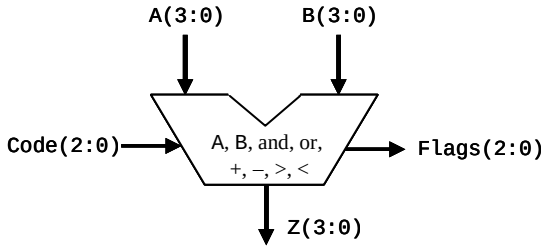


Figura 4-3: Entradas y salidas del circuito "alu_peq" (3)

4.1.4 Circuito "alu_core" (4)

El cuarto circuito es el núcleo de la ALU del microprocesador 8051 distribuido por *Oregano Systems* [101]. Este circuito es sencillo, ya que en el núcleo de esta ALU sólo hay operaciones lógicas. En la figura 4-4 se muestran las entradas y salidas.

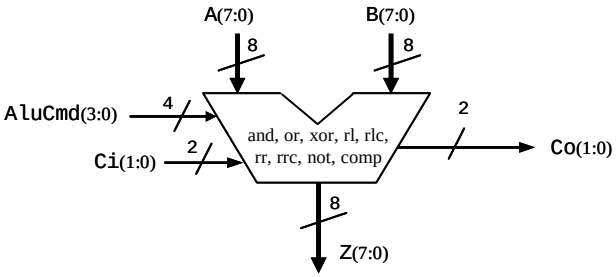


Figura 4-4: Entradas y salidas del núcleo de la ALU del 8051 (circuito 4)

Aunque el ancho de bus de los operadores es genérico, para las pruebas se ha escogido un ancho de 8 bits. Este circuito tiene un mayor número de puertos que el circuito anterior (22 de entrada y 10 de salida), sin embargo, debido a la sencillez de sus operaciones y a que la mayoría de las operaciones son a nivel de bits, el circuito tiene un menor número de puertas (219).

Se incluyen 9 operaciones: AND, OR, XOR, rotación y desplazamiento a la derecha e izquierda, inversión y comparación. Las operaciones se seleccionan con la señal *AluCmd*, cuando no coincide con el código de ninguna operación todas las salidas quedan a cero.

4.1.5 Circuito "addsub" (5)

El quinto circuito es el bloque sumador-restador de la ALU del microprocesador 8051 distribuido por *Oregano Systems* [101]. Las entradas y salidas se muestran en la figura 4-5. El número de bits de los operandos depende de un genérico *n* y por tanto el rango de la suma puede modificarse.

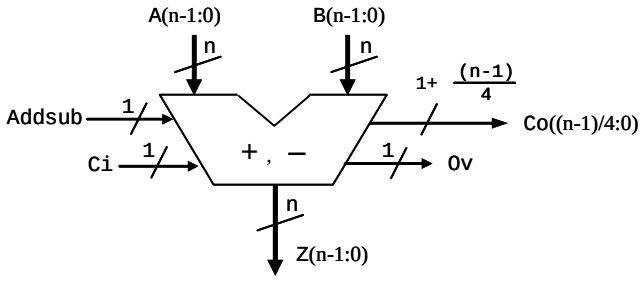


Figura 4-5: Entradas y salidas del circuito "addsub" (5)

Las entradas son:

- *Addsub* indica si se realiza suma o resta. Cuando *Addsub* = '1' se realiza la suma, cuando *Addsub* = '0', se realiza la resta haciendo el complemento a dos de *B*.
- El operando *A* es un sumando o el minuendo. El ancho del bus vale *n*.
- El operando *B* es un sumando o el sustraendo. El ancho del bus vale *n*.
- *Ci* es el acarreo de entrada.

Las salidas son:

- *Z* es el resultado de la suma o la resta. El ancho del bus vale *n*.
- *Ov* es un bit que indica si ha habido desbordamiento en la operación.
- *Co* es el acarreo de salida. El número de bits es 1+(*n*–1)/4. El bit más significativo es el bit de acarreo de salida. El resto de los bits corresponden con los acarreos parciales de grupos de cuatro bits (cuarteto o *nibble*)

4.1.6 Circuito "alu8051_simp" (6)

Este circuito une los dos circuitos anteriores mediante multiplexores y alguna lógica adicional. Tiene el sufijo *simp* para indicar que es "simplificada", y que aún no es la ALU completa del 8051. El diagrama de entradas y salidas del circuito se muestra en la figura 4-6.

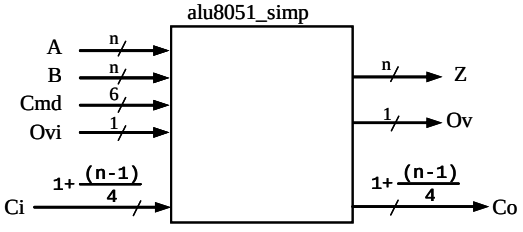


Figura 4-6: Entradas y salidas de la ALU simplificada del 8051 (circuito 6)

A continuación se muestra el esquema interno del circuito, donde aparecen el sumador-restador, el núcleo de la ALU y los bloques adicionales (figura 4-7). En el esquema de la figura se ha particularizado para un ancho de 8 bits para los operandos, sin embargo, el circuito es genérico y se puede realizar para cualquier ancho de bus.

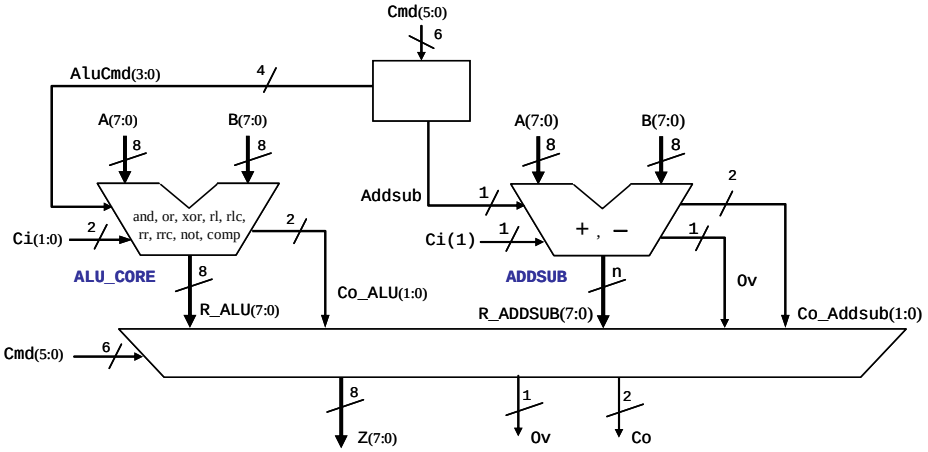


Figura 4-7: Esquema de la ALU simplificada del 8051 (circuito 6)

4.1.7 Circuito "alu8051" (7)

Este circuito ya es la ALU completa del microcontrolador 8051, aunque no incluye los bloques opcionales multiplicador, divisor y ajuste decimal. La diferencia con la ALU simplificada es que ahora los operandos se seleccionan de la ROM, la RAM y el acumulador. El esquema de entradas y salidas se muestra en la figura 4-8.

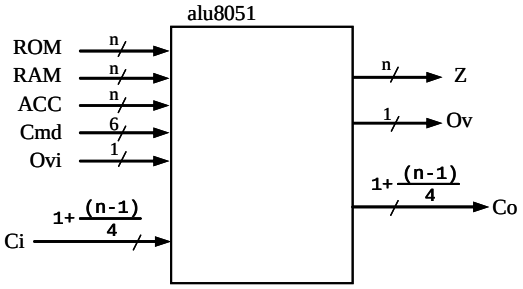


Figura 4-8: Entradas y salidas de la ALU del 8051 (circuito 7)

El esquema interno del circuito se ha dibujado en la figura 4-9.

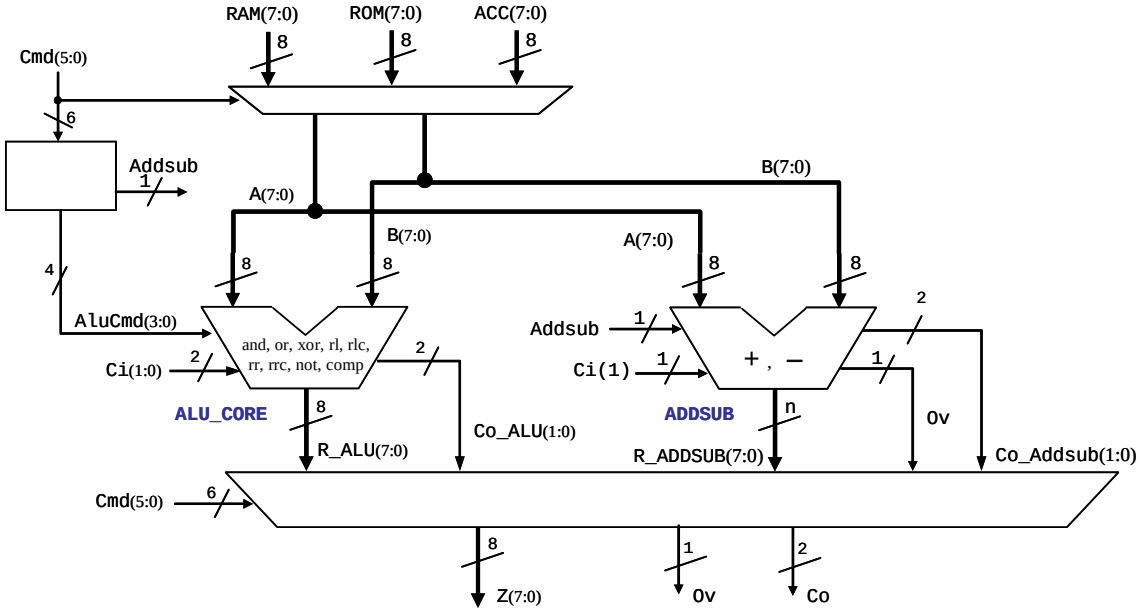


Figura 4-9: Esquema de la ALU del 8051 (circuito 7)

4.2 Análisis de los modelos por circuito

Como la herramienta que se ha creado no acepta todo el conjunto sintetizable del VHDL [60] los circuitos han tenido que ser adaptados. El análisis de probabilidades se ha realizado bit a bit aunque las señales sean vectores. Sin embargo, en los análisis se han utilizado los índices de los vectores para obtener órdenes adecuados.

Para cada circuito se compara el tamaño de los BDD resultantes de tres análisis distintos:

- Circuito en puertas (*puertas*)
- Circuito en RTL sin partición por regiones disjuntas (*RTL* o *RTL-exacto*)
- Circuito en RTL con partición por regiones disjuntas (*RTL-disjunto*)

En los tres análisis se realiza la partición por regiones reconvergentes, y sólo en el último se realiza además la partición por regiones disjuntas.

Para simplificar la denominación, los análisis se nombrarán: *puertas*, *RTL* (o *RTL-exacto*) y *RTL-disjunto*. Así que cuando se hable de "los resultados en RTL" se entenderá que es sin partición en regiones disjuntas, aunque sí con regiones reconvergentes. Cuando se especifique "resultados en RTL-disjunto" se entenderá que se ha hecho la partición en regiones disjuntas (y previamente en regiones reconvergentes §3.4).

Los resultados de probabilidad y actividad de los dos primeros análisis son idénticos. Por tanto sería lógico tomar aquel que consuma menos recursos y sea más rápido. Para el tercer análisis, que no es exacto, habrá que estudiar si compensa el error cometido con los beneficios de procesamiento, uso de memoria y rapidez.

Para las comparaciones de los tamaños se mostrarán dos grupos de columnas[46]:

- El primer grupo de columnas de resultados se llama **Total** y contiene 4 columnas. En estas columnas se indica la suma de los nodos de los BDD necesarios para el cálculo de las probabilidades y actividades de los puertos de salida. Por orden de aparición, las columnas contenidas en este grupo son:

[46] Véase la tabla 4-2 de la página 131 como ejemplo

1. La primera columna de este grupo se ha llamado **Regiones**. Indica el número de señales (sin contar los puertos de entrada) cuya probabilidad y actividad se necesitan calcular para obtener las probabilidades y actividades de los puertos de salida. Esto es equivalente al número de particiones que se realizan (por eso se llama *regiones*, indicando el número de regiones en las que se parte el circuito). Si un circuito tiene el mismo número de puertos de salida que de regiones significa que no se ha podido realizar ninguna partición.
2. La columna llamada **BDD** muestra la suma de los nodos de los BDD de probabilidad de todas las regiones en que se ha dividido el circuito.
3. La columna **TFBDD** indica la suma de los nodos de los TFBDD de todas las regiones del circuito.
4. La columna ***a*BDD** indica la suma de los nodos de los *a*BDD de todas las regiones del circuito.

- El otro grupo de columnas indica el tamaño del mayor BDD. Normalmente, este BDD se corresponderá con la partición de mayor tamaño. Consecuentemente, dentro de este grupo de columnas no figura la columna de *regiones*, ya que la información de este grupo se refiere a una sola región. Las tres columnas que contiene este grupo indican el número de nodos de cada tipo de BDD. No siempre la región con el mayor BDD de probabilidad tiene los mayores BDD de actividad. En este caso se tomarán los mayores BDD aunque no sean de la misma región.

A continuación se procede a exponer los resultados por cada circuito.

4.2.1 Circuito "max"

En la tabla 4-2 se muestran los tamaños de los BDD del circuito "max" para cuatro bits de ancho de bus. En este circuito, el análisis en el nivel RT sin realizar partición por regiones disjuntas (§3.4) proporciona las mismas cuatro regiones de reconvergencia que para el circuito en puertas. Nótese que el circuito tiene 4 puertos de salida, esto significa que la región reconvergente cubre todo el circuito, y por tanto, no se realiza partición La versión del circuito en puertas es la proporcionada por la misma referencia (Navabi [144]).

ancho de bus: 4 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas	4	74	1118	712	44	861	548
RTL	4	29	160	109	12	72	50
RTL-disjunto	7	38	193	121	6	32	19

Tabla 4-2: Tamaños de los BDD del circuito "max" (circuito 1) según el análisis realizado

En la tabla se puede apreciar que a pesar de que en puertas y en RTL las regiones son del mismo tamaño, debido al mejor ordenamiento RTL de los BDD (§3.6), **los tamaños en RTL son considerablemente menores**. Esto es especialmente acusado para el BDD de mayor tamaño, donde los BDD de actividad son más de diez veces mayores en puertas que en RTL (TFBDD: 861→72 ; *a*BDD: 548→50).

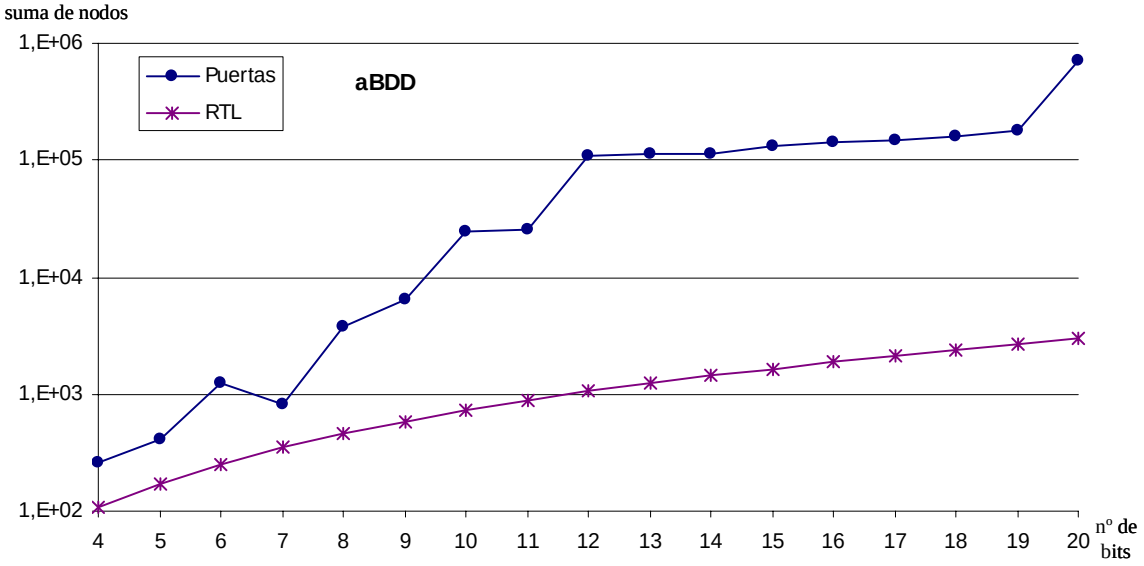
En este circuito se tiene una operación de comparación (*mayor que*) entre dos operandos de varios bits. En RTL para el ordenamiento del BDD se han tomado primero los bits más significativos, dejando para el final los menos significativos. Esto es: *A*(3), *B*(3), *A*(2), *B*(2), *A*(1), *B*(1), *A*(0), *B*(0). Identificar la operación (*mayor que*) y el índice de los vectores es una tarea inmediata desde RTL y con el EHM (§3.3.2), mientras que realizar esta identificación en el nivel de puertas es una tarea muy compleja. Por lo tanto, para este circuito, a deferencia del nivel de puertas, **en RTL resulta muy sencillo llevar un ordenamiento eficiente del BDD**.

En consecuencia, debido a que los resultados de la estimación son los mismos para puertas que en RTL (sin partición en regiones disjuntas), para este ejemplo, **es mucho más ventajoso realizar el análisis en RTL**.

Para RTL-disjunto, la suma de los nodos es mayor que en RTL (sin partición por regiones disjuntas). Esto puede parecer extraño, y se debe a que al haber más regiones y de muy pequeño tamaño, hay más variables de BDD y hay menos posibilidad de simplificación de estructuras BDD. Y es que, como se verá en los siguientes ejemplos y en el análisis del error del circuito (§4.3.1.1 y anexo AIII.1.2), **no conviene realizar particiones disjuntas exhaustivas** (§3.4.3 y apartado AIII.3 del anexo). Es mejor realizar el mínimo de estas particiones que proporcionen tamaños de BDD controlados.

Se ha analizado la variación del tamaño de los BDD en función del ancho de bus de los operandos. Para ello se ha modificado el circuito RTL original variando el ancho de bus de los operandos de entrada y se ha sintetizado con el *Design Compiler™* de *Synopsys®* [125]. La síntesis se ha realizado sólo con puertas de una y dos entradas. Y se han comparado los tamaños de los BDD en RTL sin partición por regiones disjuntas con los tamaños de los BDD en puertas. En el apartado 3.6 se ha explicado la capacidad del nivel RT de lograr un buen orden de las variables del BDD, además, en este caso, se cuenta con la información RTL del tipo de operación: un "mayor que" dentro de la condición. Por el contrario, en puertas el orden es bastante aleatorio ya que depende de la disposición de las puertas que realice el sintetizador.

En la gráfica 4-1 se muestra la evolución de los tamaños de los *a*BDD con el número de bits para el análisis en puertas y en RTL. Nótese que, debido a las grandes diferencias, la gráfica está en escala logarítmica.



Gráfica 4-1: Evolución del tamaño de los aBDD con el número de bits, circuito "max" (1)

En esta gráfica se puede apreciar la variabilidad de los *a*BDD resultantes del análisis en puertas, y lo predecible que resultan los *a*BDD del análisis en RTL. Los datos numéricos de este análisis se han incluido en la tabla AIII-1 del apartado AIII.1.1 del anexo, en donde además se incluye un análisis más amplio.

Analizando los datos numéricos de la tabla AIII-1 del anexo se extrae que, en RTL, el tamaño del mayor BDD (peor caso) sigue una progresión aritmética con el número de bits. Cuyos términos generales son $3 \cdot n$ para los BDD de probabilidad; $21 \cdot (n-1)+9$ para los TFBDD y $15 \cdot (n-1)+5$ para los *a*BDD.

Además, tanto para los BDD de probabilidad como de actividad, la suma de los nodos de todos los BDD se obtiene realizando la suma de los elementos de la sucesión de los peores casos para todos los bits menores e iguales que n. Esto es, la suma total de nodos para n bits es la suma total de nodos para $n-1$ bits más el número de nodos del peor caso para n bits.

Por tanto, al contrario con lo que ocurre en el nivel de puertas, **con el orden RTL, el tamaño de los BDD sigue una progresión lineal** que es perfectamente abordable. Por ejemplo, el mayor *a*BDD para 64 bits tendrá 950 nodos, muchos menos que los obtenidos en puertas para 20 bits: 181869. Y así, cuando se tiene un crecimiento de los BDD de actividad tan limitado como el de este ejemplo, no es preciso recurrir a la partición por regiones disjuntas.

En el apartado AIII.1.1 del anexo se ha incluido un estudio más amplio acerca la variabilidad de los tamaños de los BDD obtenidos en puertas. En este anexo se muestra que **algunos análisis en el nivel de puertas no se han podido realizar** por la excesiva demanda de memoria de los BDD resultantes. Lo que demuestra las ventajas del procesamiento RTL frente a puertas.

Para resumir, en este circuito se ha visto una de las **ventajas del procesamiento RTL frente a puertas**. En este circuito el ordenamiento RTL de los BDD es tan beneficioso que ni siquiera es necesario recurrir a realizar la partición por regiones disjuntas y por lo tanto, no se produce error en la estimación. El ordenamiento RTL logra un aumento lineal del tamaño de los BDD con el número de bits de los operandos. Este aumento lineal es muy diferente a los tamaños de los BDD obtenidos con el análisis en puertas, que no siguen una progresión aritmética y además a veces se obtienen tamaños de BDD tan grandes que imposibilitan la estimación.

4.2.2 Circuito "comparador"

Al contrario que el circuito anterior, este circuito tiene poca reconvergencia. De hecho, **el análisis del diseño RTL realiza la partición en regiones de reconvergencia** (§3.4) **que coincide con la partición en regiones disjuntas** (§3.4.1). Como consecuencia, **en ningún caso se comete error al realizar el análisis en el nivel RT**.

Cabría esperarse que para el circuito en nivel de puertas se hiciese una partición similar, sin embargo, **en el circuito sintetizado en puertas lógicas no se puede realizar ninguna partición**. El motivo de que no se pueda realizar la partición es que **la síntesis ha dispuesto las puertas de manera que han aumentado las reconvergencias**.

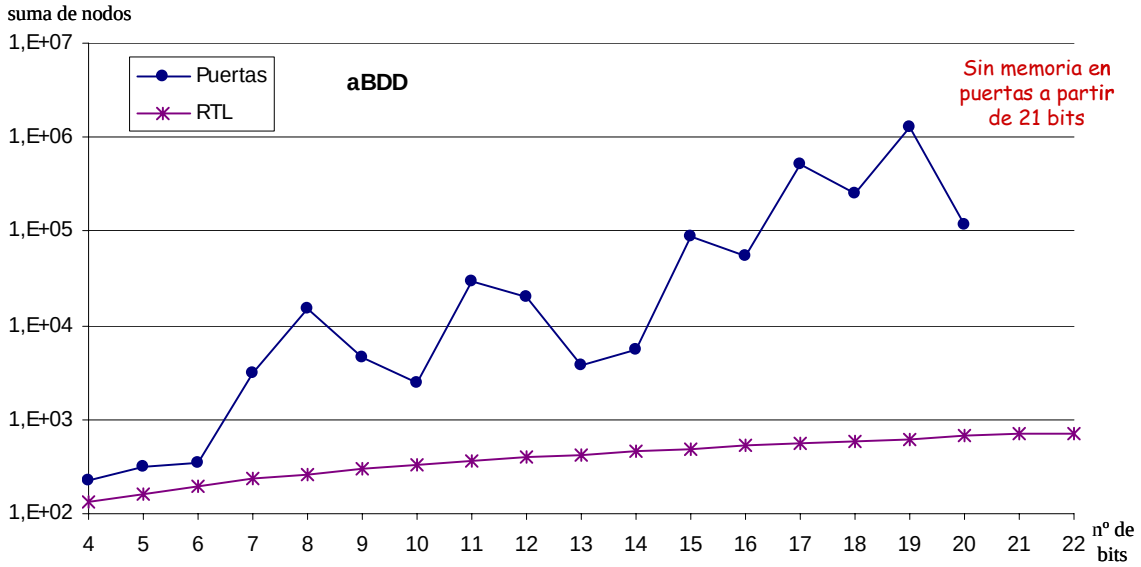
En la tabla 4-3 se muestran los tamaños de los BDD. En este caso, para RTL y RTL-disjunto el resultado es idéntico porque como se ha dicho, la partición es la misma. En el apartado AIII.2 del anexo se amplía la información sobre la partición reconvergente realizada.

ancho de bus: 4 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas	3	51	329	230	19	135	96
RTL	12	48	204	132	4	18	12
RTL-disjunto	12	48	204	132	4	18	12

Tabla 4-3: Tamaños de los BDD del circuito "comparador" (circuito 2) según el análisis realizado

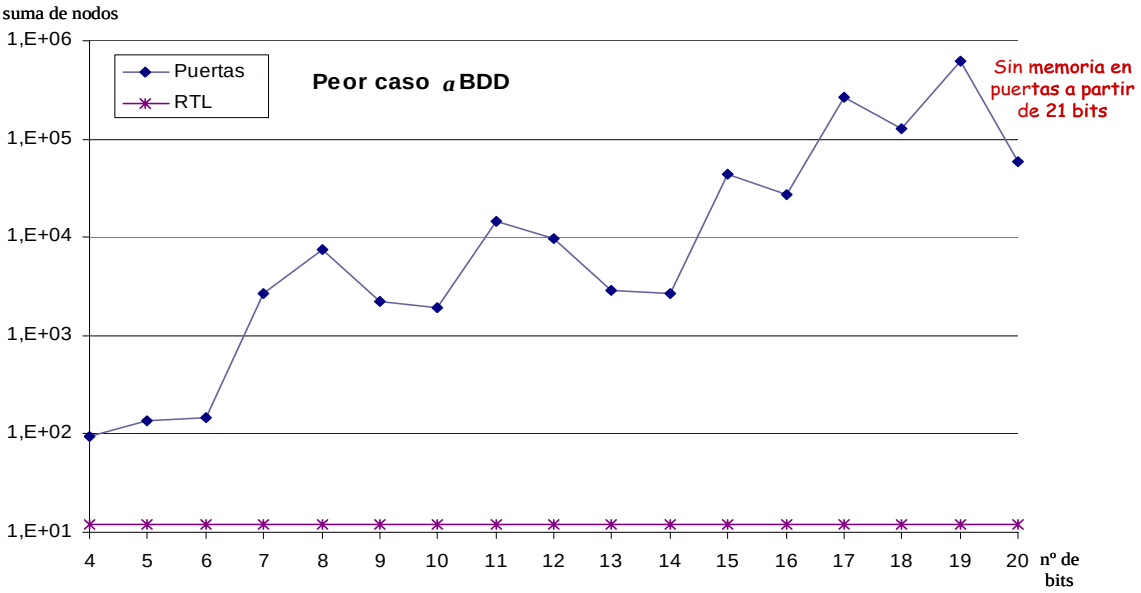
Analizando la tabla se observa que en puertas el tamaño del BDD de probabilidad es similar al del RTL. Esto, como ya se ha comentado, es porque se pueden simplificar partes de un BDD con otro, además de que al haber menos particiones no existen los nodos intermedios que los unen. Sin embargo, los BDD de actividad en RTL son considerablemente menores que en puertas. Estas diferencias aumentan con el ancho de bus de los operandos.

En la gráfica 4-2 se muestra el incremento de la suma de nodos los *a*BDD del circuito según se realice el análisis en puertas o en RTL. Nótese que la escala es logarítmica.



Gráfica 4-2: Evolución del tamaño de los aBDD con el número de bits, circuito "comparador" (2)

La evolución de los *a*BDD de mayor tamaño se muestra en la gráfica 4-3.



Gráfica 4-3: Evolución del tamaño del mayor aBDD con el número de bits, circuito "comparador" (2)

Los datos numéricos de las gráficas 4-2 y 4-3 se encuentran en la tabla AIII-4 del apartado AIII.2. De estas gráficas se extrae que:

- El aumento del número de nodos con el ancho de bus de los operandos es lineal para el nivel RT, mientras que en puertas no sólo no es lineal sino que resulta imprevisible
- Llegado a un tamaño del circuito (21 bits), en puertas no se puede analizar debido al gran tamaño de sus BDD
- En RTL, el tamaño del mayor BDD (peor caso) es constante aunque se varíe el ancho de bus de los operandos (BDD: 4, TFBDD: 18, *a*BDD: 12). Mientras que en puertas es muy variable y, por lo general, aumenta considerablemente con el número de bits.

Por tanto, gracias a que en RTL se puede realizar la partición reconvergente, y que por esto se mantienen constantes los BDD de mayor tamaño, este circuito se puede abordar en RTL para cualquier número de bits, ya que nunca habrá excesivas demandas de memoria. En RTL el número de nodos totales sigue una progresión aritmética con el número de bits de los operandos (*n*). Esta progresión es diferente según el tipo de BDD:

- Número de nodos totales para los BDD: $12 \cdot n$
- Número de nodos totales para los TFBDD: $51 \cdot n$
- Número de nodos totales para los *a*BDD: $33 \cdot n$

Se debe destacar la importancia de lograr un aumento lineal del número de nodos del circuito y de que los BDD de mayor tamaño permanezcan constantes. Por otro lado, se observa la reducción de un 35% en el número de nodos de los *a*BDD propuesto en esta tesis frente a los TFBDD.

Como se ha comentado, en puertas es más difícil obtener un orden de variables óptimo, ya que debido a la poca información disponible de alto nivel, es difícil tomar decisiones en cuanto al orden de las variables en el BDD. En los análisis aquí realizados se ha tomado el orden de aparición en la descripción en puertas resultante de la síntesis. Si bien este orden de aparición puede tener algún significado estructural, es bastante aleatorio. En el anexo AIII.2 se muestra los tamaños de los BDD para el circuito en puertas con otros órdenes (véase también la tabla AIII-1).

La conclusión más importante de este circuito es la constatación de que **la síntesis de RTL a puertas puede aumentar la reconvergencia de un circuito, disminuyendo el número de particiones reconvergentes** que se pueden realizar. Esto hace que aumente el tamaño de las regiones del circuito en puertas, pudiendo provocar incluso que el circuito no se pueda analizar en el nivel de puertas por el excesivo tamaño de sus BDD. Es por lo tanto, **una de las ventajas del análisis en RTL**.

4.2.3 Circuito "alu_peq"

Este circuito es bastante más grande que los anteriores, y por lo tanto, las diferencias son más apreciables (si se comparan manteniendo el mismo ancho de bus de 4 bits)

En la tabla 4-4 se muestra el número de particiones y las diferencias de tamaños de los BDD, para un ancho de bus de 4 bits.

ancho de bus: 4 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas	17	386	8999	5667	108	3958	2512
RTL	29	260	2171	1394	55	637	438
RTL-disjunto	31	251	1539	946	15	157	85

Tabla 4-4: Tamaños de los BDD del circuito "alu_peq" (circuito 3)

En primer lugar se observa que en el nivel de puertas hay menos particiones, y por lo tanto, tienen mayor tamaño. Igual que en el ejemplo anterior, esto se debe a que **la síntesis ha aumentando la reconvergencia**. El número de nodos de los BDD de actividad resultan cuatro veces más pequeños en RTL que en puertas, y si se realiza la partición por regiones disjuntas aproximadamente salen seis veces menor (aunque para el tamaño de este circuito no sea necesario).

Más acusada es la diferencia con el peor caso, en RTL los BDD de actividad son unas seis veces menor que en puertas, y si se aplica la partición por regiones disjuntas es entre 25 y 30 veces menor. Para los BDD de probabilidad la diferencia es de la mitad para RTL y con regiones disjuntas unas siete veces menor.

En el anexo AIII.3 se amplía información sobre otros casos de este circuito.

4.2.4 Circuito "alu_core"

La mayoría de las operaciones de este circuito se realizan a nivel de bits, por tanto, son operaciones de baja complejidad. Aunque este circuito tiene más puertos de entrada que el circuito anterior, por ser un circuito más pequeño y por involucrar menos bits en las operaciones, los BDD resultan menores. En la tabla 4-5 se muestran los tamaños de los BDD según el análisis realizado.

ancho de bus: 8 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas	25	423	5010	3328	45	639	427
RTL	26	263	2472	1547	23	277	168
RTL-disjunto exhaustivo	50	303	2448	1495	21	250	149
RTL-disjunto mínimo	26	263	2392	1519	23	264	163

Tabla 4-5: Tamaños de los BDD del circuito "alu_core" (circuito 4)

Analizando la tabla se ve que en puertas los tamaños de los BDD son mayores, así como también lo es su BDD de mayor tamaño. Para RTL y RTL-disjunto los resultados son similares, tanto si se realiza la partición disjunta exhaustiva como la mínima. Se debe señalar que, aunque el número de particiones para el análisis RTL y RTL-disjunto mínimo coincidan, no son las mismas particiones.

Por la poca diferencia entre los tamaños de los BDD en RTL y RTL-disjunto, para este ejemplo no es necesario realizar la partición por regiones disjuntas. Esto se debe a que la mayoría de las operaciones son a nivel de bits, y es por esto que los BDD no aumentan sustancialmente de tamaño con el ancho de bus de los operandos.

La necesidad de realizar la partición por regiones disjuntas se puede detectar durante el mismo proceso de particionado. Como se dispone del EHM (§3.3.2), éste proporciona la información relativa al número de dependencias de la región y de la complejidad de las operaciones involucradas, y así anticipar la necesidad de realizar la partición por regiones disjuntas.

Aún así, se verá en el apartado del análisis del error (§4.3.1.4 y AIII.4) que realizando la partición disjunta mínima el error es despreciable, y que el error en la partición disjunta exhaustiva solamente es importante en ciertos casos que no son reales durante el funcionamiento normal del circuito.

4.2.5 Circuito "addsub"

Para 8 bits de ancho de bus, a pesar este circuito tiene menor número de puertas que los dos anteriores (ver tabla 4-1) los BDD salen mayores para este sumador/restador. Esto se debe principalmente al mayor número de puertas XOR que tienen los sumadores, y que estas puertas producen BDD de gran tamaño. En la tabla 4-6 se muestran los tamaños de los BDD para el circuito de 8 bits de ancho de bus. Una ampliación de esta tabla para diferente número de bits se ha añadido al anexo (tabla AIII-14).

ancho de bus: 8 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas	13	571	11177	8249	139	3837	2856
RTL	11	499	3599	2590	89	641	479
RTL-disjunto	45	187	645	375	5	19	13

Tabla 4-6: Tamaños de los BDD del circuito "addsub" (circuito 5) con 8 bits de bus de datos

En este circuito hay una gran diferencia entre los tres análisis realizados. El circuito no se puede dividir en regiones reconvergentes. En la tabla se puede apreciar que resultan 11 regiones en

RTL y 13 en puertas (ya que en puertas hay dos pequeñas regiones más). Como hay 11 puertos de salida, para cada uno de ellos se emplea una única región que va desde el puerto hasta las entradas.

Sin embargo, si se realiza la partición por regiones disjuntas, se obtienen 45 regiones. Esta partición hace que la suma de los nodos de los BDD sea mucho menor, y sobre todo el BDD de mayor tamaño (peor caso). En el apartado AIII.5.1 del anexo se muestran detalles de la partición de un sumador/restador.

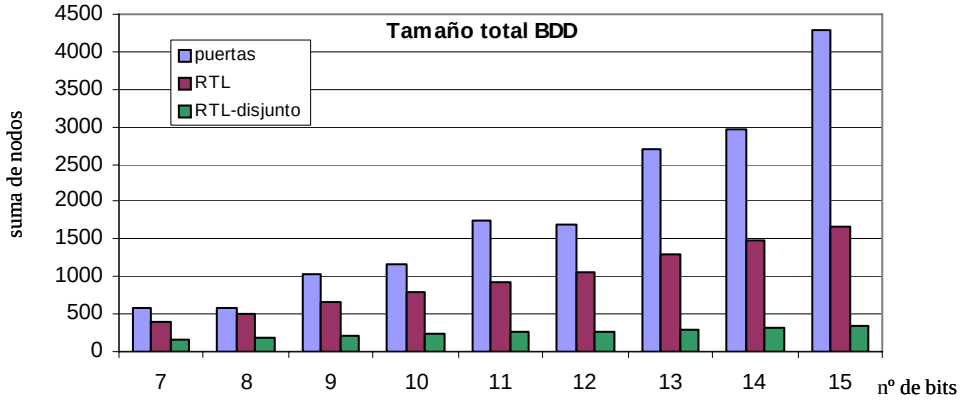
La suma de los tamaños de los BDD de probabilidad en puertas es tres veces mayor que la de los de RTL-disjunto. Los de actividad en puertas son unas 20 veces mayor. Más diferencia hay para el peor caso.

Cuando en RTL no se realiza partición por regiones disjuntas, el tamaño de los BDD es también bastante mayor que con partición, pero se mantienen considerablemente menores que en puertas. Esto se debe al mejor ordenamiento que se logra en RTL, ya que se compara entre regiones de tamaño similar (incluso ligeramente mayores en RTL).

Obsérvese la diferencia entre el tamaño de los TFBDD en análisis en puertas que suman 11177 nodos con el tamaño de los *a*BDD propuestos en esta tesis y realizando el análisis en RTL-disjunto que suman 375 nodos (casi 30 veces menor).

Esta diferencia es aún mayor para el peor caso, en donde se pasa de 3837 a 13 nodos (295 veces menor). Este cálculo es aproximado y el análisis del error se realiza en el apartado 4.3.1.5.

Para este circuito se han analizado los tamaños de los **BDD de probabilidad** para diferente número de bits. En la gráfica 4-4 se muestra la evolución de los tamaños de los BDD según el número de bits. En el anexo AIII.5 se muestran los datos numéricos de estas gráficas.



Gráfica 4-4: Evolución del tamaño de los BDD de probabilidad con el número de bits

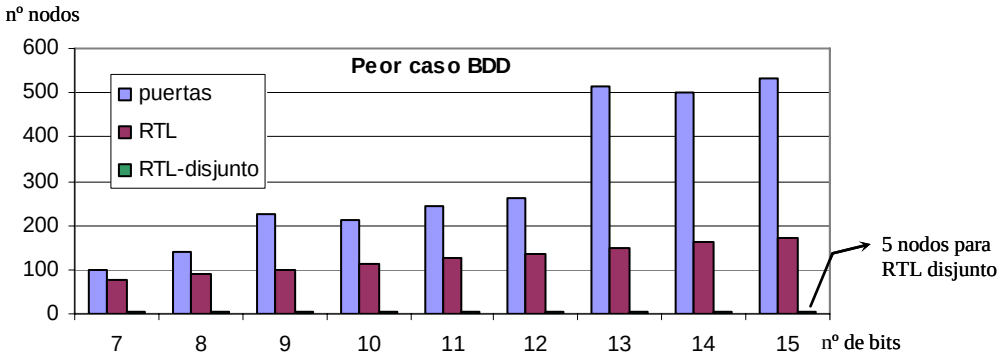
En las gráficas se muestran tres barras:

- La primera, llamada "puertas" se corresponde con el análisis en puertas.
- La segunda, llamada "RTL" muestra el análisis en RTL con partición en regiones reconvergentes, pero sin usar las regiones disjuntas (se trata del RTL-exacto). En este caso, el circuito no es divisible en regiones reconvergentes.
- La última barra ("RTL-disjunto") se corresponde con el análisis en RTL realizando partición por regiones disjuntas. Como se ha visto, este análisis proporciona resultados aproximados para la actividad. El error que causa la partición disjunta se analizará en el apartado 4.3.1.5.

A partir de la gráfica (y de las tablas del anexo AIII.5) se observa que el tamaño de la suma de los nodos de los BDD **aumenta linealmente para el análisis RTL disjunto** (la tercera barra). Sin embargo, mediante el análisis en puertas, el incremento no es lineal, y sufre variaciones poco previsibles. El análisis en RTL tiene un incremento más acusado que en RTL-disjunto pero bastante menor que en puertas. Además, el incremento en RTL es más constante que en puertas,

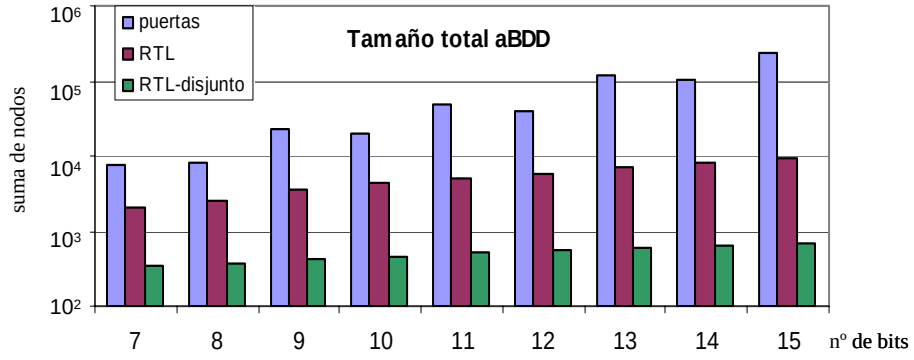
donde existe una mayor variabilidad (se observa mayor incremento cuando hay un número de bits impar).

En la gráfica 4-5 se muestra la evolución del mayor BDD de probabilidad con el número de bits. Este BDD de mayor tamaño no cambia para el análisis RTL disjunto, manteniéndose en 5 nodos. Para RTL el tamaño del peor caso sigue una progresión aritmética cuya diferencia es de 12 nodos. Por el contrario, para el aumento del tamaño en el análisis en puertas muy variable, pudiendo existir grandes diferencias. La poca homogeneidad de este aumento en el análisis en puertas se explica por la dependencia del orden de las variables en el tamaño del BDD. En consecuencia, debido a que en puertas se escoge un orden aleatorio, el tamaño de los BDD no tiene un aumento proporcional.



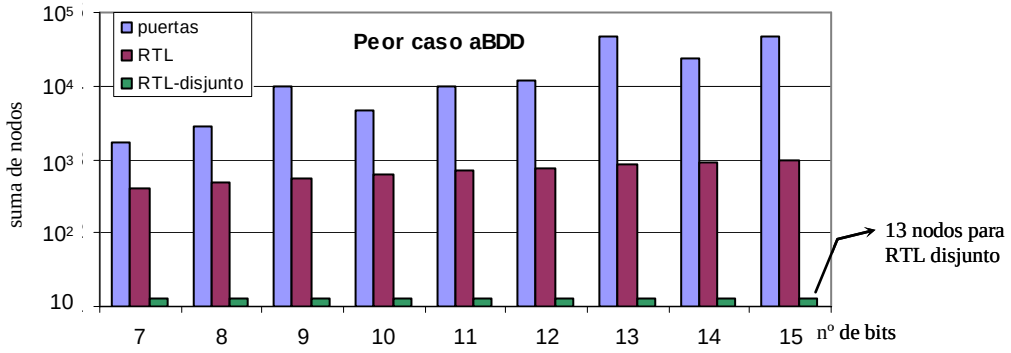
Gráfica 4-5: Evolución del tamaño del mayor BDD de probabilidad con el número de bits

El análisis del incremento de los tamaños de los para los **BDD de actividad** muestra todavía más diferencias. En la gráfica 4-6 se muestra el aumento de la suma del número de nodos de los *a*BDD del circuito con el ancho de bus. Nótese que la escala es **logarítmica**.



Gráfica 4-6: Evolución del tamaño de los aBDD con el número de bits

Para el *a*BDD de mayor tamaño, éste aumenta en los casos que no se realiza partición. En el análisis en puertas el aumento es imprevisible mientras que en RTL aumenta de forma lineal con un incremento de 74 nodos por cada bit. Cuando se divide el circuito en regiones disjuntas, éste permanece constante en 13 nodos. En RTL el aumento es lineal, y en puertas el aumento es imprevisible. En la gráfica 4-7 se muestra la evolución. Nótese otra vez que esta gráfica también está en escala logarítmica.



Gráfica 4-7: Evolución del tamaño del mayor aBDD con el número de bits

Al realizar el análisis variando el número de bits se evidencian las ventajas realizar el análisis en RTL, tanto por el mejor ordenamiento que se consigue, como por la posibilidad de realizar particiones disjuntas.

4.2.6 Circuito "alu8051_simp"

Este circuito y el siguiente destacan sobre los anteriores por su mayor tamaño y principalmente por su mayor número de entradas. Debido al gran aumento del tamaño de los BDD con el número de variables, resulta fundamental poder dividir el circuito. Sin embargo, en estos circuitos no se logran buenas particiones por regiones reconvergentes. En RTL no se logra realizar ninguna partición por regiones reconvergentes: el número de regiones coincide con el número de puertos de salida. En puertas se puede dividir en más regiones, pero sigue habiendo regiones de gran tamaño. Consecuentemente, si se quiere realizar partición se debe recurrir a la partición por regiones disjuntas.

Para un ancho de bus de 8 bit, la suma del número de nodos de los BDD y el peor caso se muestran en la tabla 4-7.

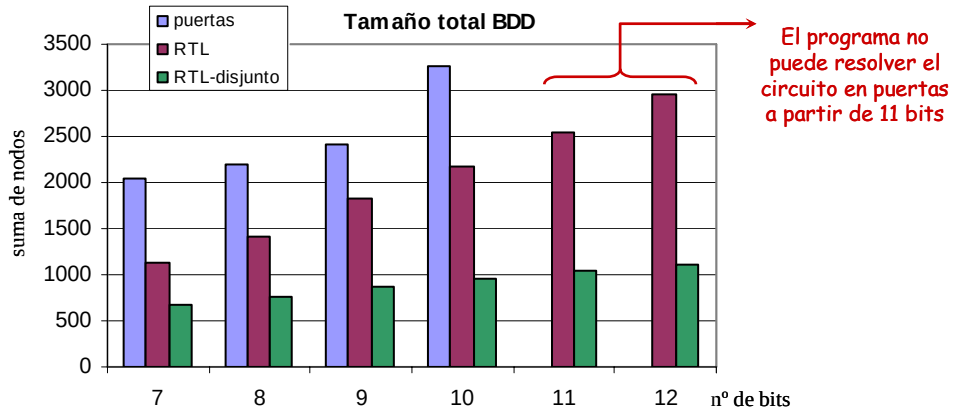
ancho de bus: 8 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas	21	2198	273365	198584	908	237571	175896
RTL	11	1423	26061	16813	224	4881	3215
RTL-disjunto	123	765	7068	4134	34	645	370

Tabla 4-7: Tamaños de los BDD del circuito "alu8051_simp" (circuito 6) con 8 bits de bus de datos

En esta tabla se aprecian grandes diferencias entre los tamaños de los BDD, especialmente para los de actividad. Realizando el análisis en RTL se consigue dividir por diez el número de nodos de los BDD de actividad. Siendo el peor caso unas 50 veces menor en RTL. Reducciones aún más grandes se consiguen partiendo el circuito en regiones disjuntas, aunque, como ya se ha dicho, este procedimiento no es exacto. En el apartado 4.3.1.6 se analiza el error cometido.

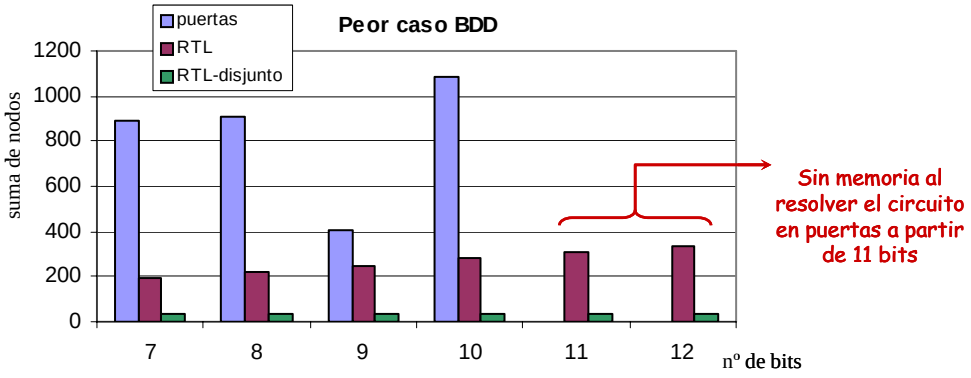
En RTL, el orden seguido para los BDD ha sido tomar primero la señal de selección (§3.6) y posteriormente los operandos, tomando primero los más significativos. De manera similar al circuito "max", en RTL y con el EHM es muy fácil de obtener el orden, mientras que en puertas se convertiría en una tarea muy compleja.

La evolución del tamaño de los **BDD de probabilidad** se muestra en la gráfica 4-8. En la gráfica no aparecen los tamaños de los BDD de probabilidad para el circuito en puertas de 11 y 12 bits debido a que la computadora se quedaba sin memoria al realizar los cálculos.



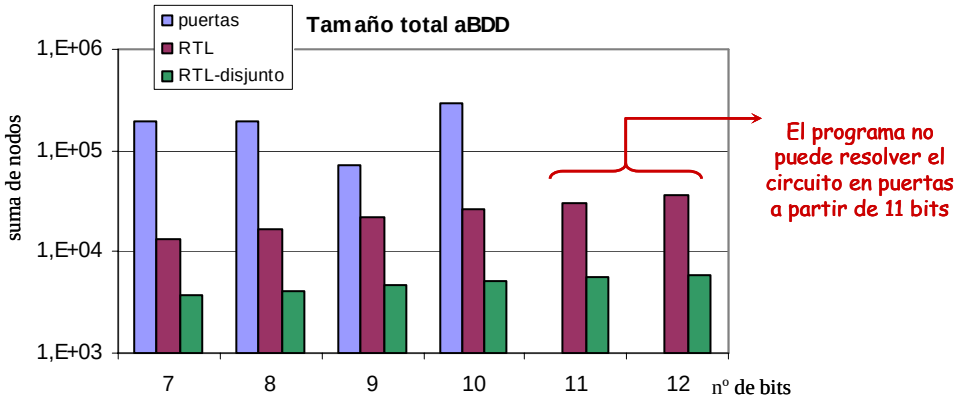
Gráfica 4-8: Evolución del tamaño de los BDD de probabilidad con el número de bits

Seguidamente, en la gráfica 4-9 se muestra la evolución para el BDD de probabilidad de mayor tamaño. Nótese que en puertas la evolución es variable debido a la aleatoriedad del ordenamiento de los BDD. Debido a esto, para nueve bits hay una disminución en el tamaño. El BDD de mayor tamaño en el análisis RTL-disjunto se mantiene constante en peor caso en 34 nodos para cualquier ancho de bus.



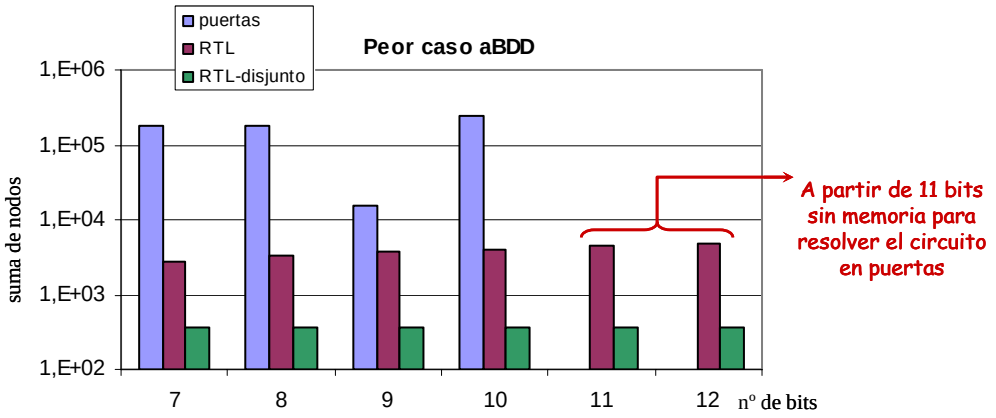
Gráfica 4-9: Evolución del tamaño del mayor BDD de probabilidad con el número de bits

El análisis de los tamaños de los **BDD de actividad** aún muestra mayores diferencias. Para los *a*BDD, la gráfica 4-10 muestra la evolución. Nótese que la escala es logarítmica.



Gráfica 4-10: Evolución del tamaño de los aBDD con el número de bits

Y la evolución del *a*BDD de mayor tamaño se muestra en la gráfica 4-11. El tamaño del mayor BDD de actividad en el análisis RTL-disjunto se mantiene constante, siendo de 645 nodos para los TFBDD y 370 si se usan los *a*BDD propuestos en esta tesis. En RTL, el aumento del *a*BDD de mayor tamaño sigue una progresión aritmética de diferencia 423 (para TFBDD la diferencia es 609). Mientras que en puertas el aumento de tamaño no es lineal y es imprevisible.



Gráfica 4-11: Evolución del tamaño del mayor aBDD con el número de bits

En el apartado AIII.6 del anexo se muestran los datos numéricos relativos a estas gráficas.

Por tanto, el análisis realizado en este circuito **evidencia la ventaja de la estimación en RTL frente a puertas**, ya que, a partir de un determinado número de bits, **el aumento del tamaño de los BDD en puertas imposibilita el cálculo** para los circuitos en este nivel de descripción.

A raíz de estos análisis también salen a la luz las **ventajas de la partición en regiones disjuntas**. Éstas no sólo **reducen considerablemente el tamaño de los BDD**, sino que **hacen que los BDD de mayor tamaño no aumenten de tamaño con el número de bits de los operandos**. Esto **permite calcular la actividad del circuito con cualquier ancho de bus**.

4.2.7 Circuito "alu8051"

Este circuito, además de ser el más complejo, es el que más puertos de entrada tiene, y por ello es el que producirá mayores tamaños de BDD si no se logran realizar particiones. Este circuito no se puede dividir por regiones reconvergentes, ni en puertas ni en RTL. Sin embargo, dependiendo de la descripción RTL o si se realiza un análisis previo, este circuito se puede dividir en regiones disjuntas (ver anexo AIII.7.1).

En la tabla 4-8 se muestran las particiones y los tamaños de los BDD para el circuito con 8 bits de dato. Para el circuito en puertas y el análisis RTL el diseño tiene 11 regiones, y debido a que éste es el número de puertos de salida, implica que no se divide en absoluto. Para RTL disjunto el circuito se puede realizar una partición disjunta exhaustiva (§3.4.3, y anexo AIII.3.1, página 227) hasta hacer las regiones de tan sólo 5 nodos de BDD de probabilidad. Esto en realidad no es necesario, y resulta más conveniente realizar particiones de mayor tamaño, ya que, aunque los tamaños de los BDD sean mayores, se mantienen limitados y se producen menores errores (§4.3.1.7). Éste es el caso de la última fila ("RTL-disjunto 2"), donde se realiza una partición disjunta mínima (§3.4.3).

ancho de bus: 8 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas 1	11	4936	760118	467174	1176	275596	166061
Puertas 2	11	2813	162315	95937	440	28070	16178
RTL	11	2819	99679	58909	425	18262	10807
RTL-disjunto 1	596	1776	6638	3707	5	19	13
RTL-disjunto 2	289	1470	17141	9290	57	1732	912

Tabla 4-8: Tamaños de los BDD del circuito "alu8051" (circuito 7) con 8 bits de bus de datos

En la tabla 4-8 se muestra el resultado para dos circuitos en puertas. Estos circuitos son producto de la síntesis del mismo diseño RTL con opciones de optimización diferentes. Para estos dos diseños el número de nodos de los BDD de actividad de uno de ellos es 4 veces mayor

que del otro. Esto apoya el carácter tan variable del análisis en puertas. Para el peor caso, el BDD de actividad es unas 10 veces mayor que para el otro.

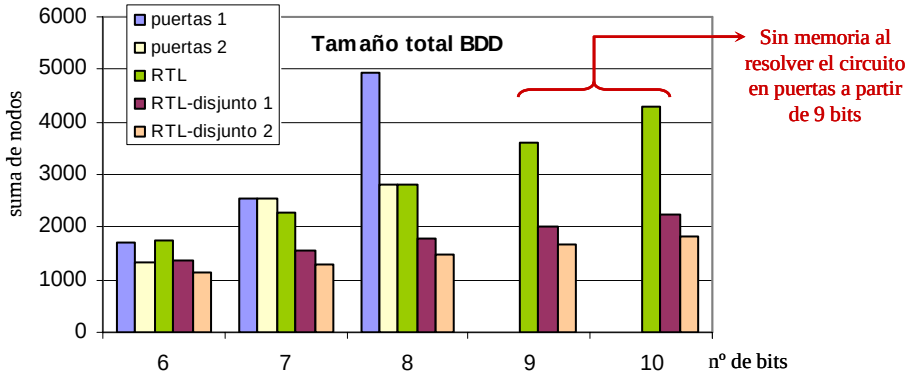
En el análisis en RTL sin partición se logran tamaños de BDD de actividad menores que los dos casos en puertas. Con partición por regiones disjuntas se logran tamaños todavía más pequeños. Se han realizado dos particiones diferentes:

- RTL-disjunto 1: realiza una partición disjunta exhaustiva, dividiendo el circuito en el máximo número de regiones (596 regiones)
- RTL-disjunto 2: realiza una partición disjunta mínima[47] (289 regiones)

Con la partición disjunta exhaustiva se obtienen tamaños de BDD muy pequeños. La suma de los nodos de los BDD de actividad es más de 15 veces menor que en RTL sin partición. Comparando los tamaños de los BDD de actividad en puertas para el primer caso (*Puertas 1*), el análisis en RTL con partición produce 100 veces menos nodos.

La segunda partición disjunta no obtiene tamaños de BDD tan pequeños como la primera. Sin embargo, los tamaños siguen siendo limitados y son menores que los resultantes en puertas y en RTL sin partición. Como se verá en el apartado 4.3.1.7, cuanto menor sea el número de particiones menor es el error cometido.

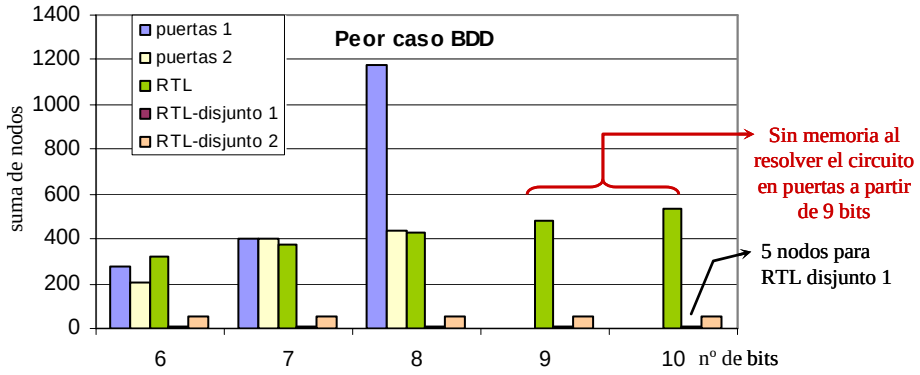
Cambiando el número de bits de los operandos, la evolución del número de nodos de los **BDD de probabilidad** se muestra en la gráfica 4-12. A partir de 9 bits el programa se queda sin memoria para el caso de puertas.



Gráfica 4-12: Evolución del tamaño de los BDD de probabilidad con el número de bits

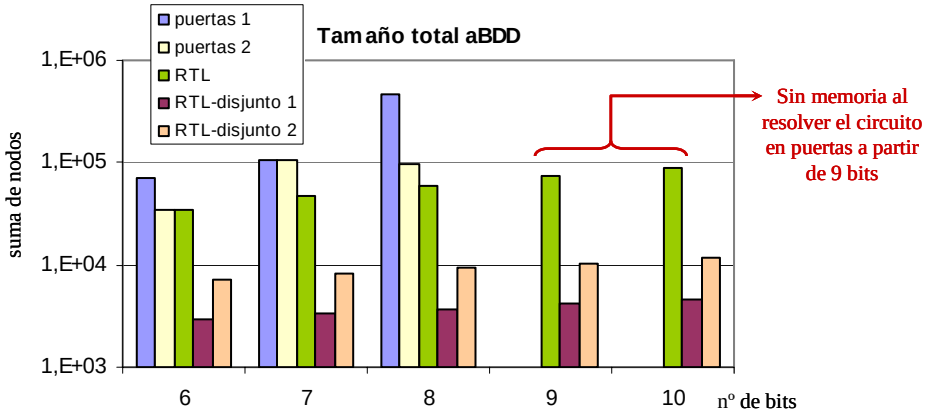
En la gráfica 4-13 se muestra la evolución del BDD de mayor tamaño. Debido a que el BDD de mayor tamaño en el análisis RTL disjunto se mantiene constante en 5 nodos, por su pequeño tamaño no se puede apreciar en la gráfica. Nótese también cómo para 6 bits la suma de los nodos del BDD en RTL es mayor que en puertas, sin embargo, a partir de 7 bits aumenta en puertas.

[47] En realidad no es mínima, porque se podrían realizar menor número de particiones, pero, como se verá en el apartado 4.3.1.7, los errores son suficientemente pequeños.



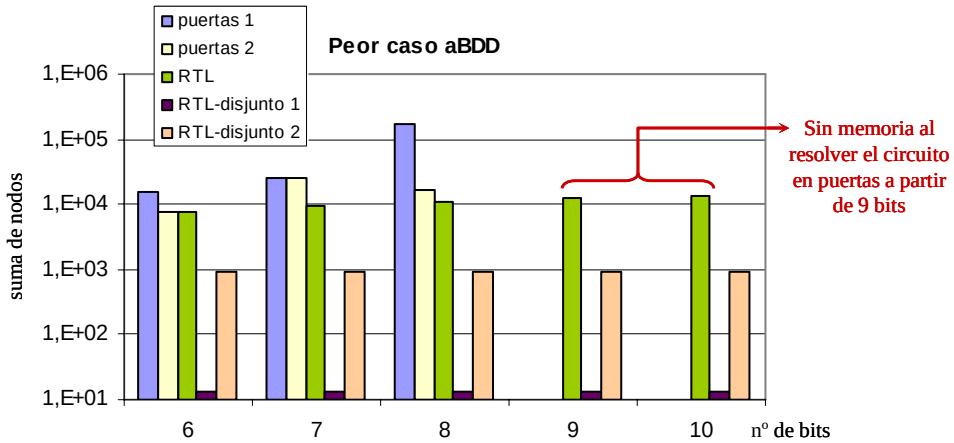
Gráfica 4-13: Evolución del tamaño del mayor BDD de probabilidad con el número de bits

El análisis de los tamaños de los **BDD de actividad** muestra mayores diferencias respecto al análisis en puertas. La gráfica 4-14 muestra esta evolución.



Gráfica 4-14: Evolución del tamaño de los aBDD con el número de bits

La gráfica 4-15 muestra la evolución del *a*BDD de mayor tamaño. El tamaño de los mayores BDD de actividad es constante con el número de bits cuando se realiza el análisis RTL-disjunto, teniendo 912 nodos en RTL-disjunto 2 y 13 nodos en RTL-disjunto 1 (para los TFBDD son 1732 y 19 nodos respectivamente) . En RTL sigue una progresión aritmética de diferencia 1467 para los *a*BDD propuestos en esta tesis y de 2416 si se usan TFBDD.



Gráfica 4-15: Evolución del tamaño del mayor aBDD con el número de bits

En la tabla AIII-17 del apartado AIII.7 del anexo (página 240) se muestran los datos numéricos relativos a estas gráficas.

Este análisis refuerza las conclusiones extraídas en el apartado anterior:

- Debido al aumento del tamaño de los BDD, normalmente, el análisis en puertas no se puede llevar a cabo a medida que crece el tamaño y la complejidad de los circuitos.
- El crecimiento del tamaño de los BDD del análisis en puertas es imprevisible
- El aumento de tamaño de los mayores BDD en RTL-exacto sigue una progresión aritmética
- El tamaño de los mayores BDD en RTL-disjunto permanece constante
- Las particiones disjuntas no sólo reducen significativamente el número total de nodos, sino que limitan el tamaño máximo del mayor BDD, lo que permite realizar el cálculo para cualquier número de bits. Para circuitos grandes y complejos, donde no se pueda realizar particiones reconvergentes, la partición disjunta puede ser necesaria para controlar el crecimiento de los BDD.

4.3 Análisis del error por circuito

En este apartado se comparan los resultados de actividad de los modelos RTL con partición y sin partición por regiones disjuntas. Ya se vio en la sección 3.4.3 que para la partición por regiones disjuntas se realizaban simplificaciones en el cálculo de la actividad, mientras que el cálculo de probabilidad era exacto. En el resto de modelos los resultados son los mismos. Esto es, **el modelo probabilístico en puertas y el modelo probabilístico en RTL sin partición en regiones disjuntas dan exactamente el mismo resultado de probabilidad y actividad,** y este resultado se considera que **es el mismo que el obtenido mediante métodos simulativos** (§2.1.1). El método simulativo no proporciona resultados exactos por su dependencia con los vectores de entrada asignados. Dos pruebas simulativas sobre el mismo circuito producen resultados ligeramente distintos. En general, estas diferencias suelen ser inferiores a 0,01 en valor absoluto. Por tanto, en esta tesis **se ha considerado que diferencias en la actividad inferiores a 0,01 son despreciables**.

Por otro lado, los cálculos realizados mediante TFBDD y *a*BDD dan los mismos resultados. Así que **el único error que hay que analizar es el cometido al realizar la partición por regiones disjuntas**.

Aún así, para los tres primeros circuitos (los más pequeños) **se han contrastado los valores de probabilidad y actividad obtenidos con el método dinámico** (o simulativo). La obtención de los resultados por métodos dinámicos se ha realizado mediante la simulación del circuito con vectores de entrada aleatorios condicionados a unas probabilidades y actividades determinadas. De la simulación se han obtenido los valores estadísticos de probabilidad y actividad. Como ya se ha indicado, los resultados de los métodos dinámicos dependen de los vectores de entrada que se hayan asignado y por tanto no son exactos a no ser que se llevasen a cabo largos tiempos de simulación.

Estos circuitos pequeños se han estimulado con tamaños de vector de 16300 ciclos. Un tamaño suficientemente grande para las dimensiones de estos circuitos y que permite no tener que recurrir a los métodos ya citados en el apartado 2.1.1 de compactación de los vectores [81], [77]. Como inconveniente están los largos tiempos de simulación. Los errores han estado por debajo de 0,01, que como se ha dicho, se deben a la dependencia de los métodos dinámicos con los vectores que se asignan. Y como ya se ha comentado, es por esto por lo que se ha tomado como 0,01 el límite a partir de el cual se considera que se ha cometido error.

A continuación se explica el método llevado a cabo en el análisis del error y posteriormente se analiza el error en cada uno de los circuitos de pruebas.

4.3.1 Análisis del error debido a las particiones disjuntas

Analizar el error no es tarea fácil ya que éste depende de los valores de actividad y probabilidad de las entradas. Por tanto hay un número infinito de combinaciones, y cada una de ellas dará un error diferente.

En general, otros estudios realizan el análisis asignando unos valores concretos para la probabilidad y actividad de las entradas. Así en [119] se asigna P_{in}=0,5 y a_{in}=0,25 para las entradas; en [28] se asigna P_{in}=0,5 y dos valores distintos de actividad: a_{in1}=0,1 ; a_{in2}=0,26 ; en otros trabajos no se especifica, pero como comparan los resultados con la referencia [119], es de suponer que lo hacen para los mismos valores.

En esta tesis se ha preferido hacer un análisis más amplio, realizando múltiples pruebas en las que se asignan los valores de una malla de valores de probabilidad y actividad. Como no todos los pares de actividad y probabilidad son posibles (§2.2.3.2, ecuación 2.22), los valores asignados deben entrar dentro del triángulo que se mostró en la figura 2-17 (y que se ha repetido aquí, en la figura 4-10). Dentro de este triángulo, se toman valores de probabilidad con saltos de 0,03125 (31 valores) y valores de actividad con saltos de 0,015625 (64 valores). En total la malla tiene 1024 puntos (dentro del triángulo), lo que hacen 1024 estimaciones diferentes de la actividad del circuito.

En la figura 4-10 se muestra la malla de valores. Dentro del triángulo hay una curva que muestra los puntos en los que no hay dependencias temporales y por tanto no se produce error. En los puntos cercanos a esta curva el error será mínimo (condición 3 del apartado 3.4.3, página 72).

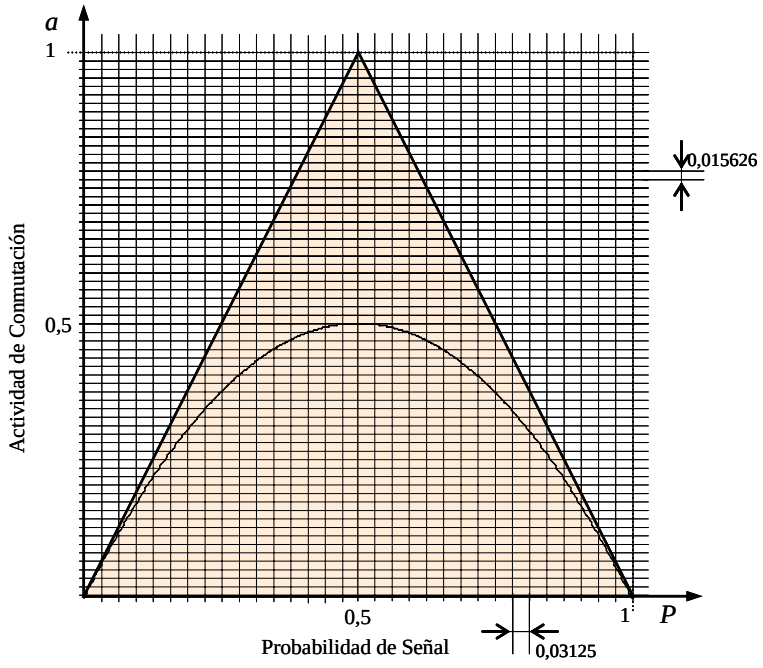


Figura 4-10: Malla de puntos que se toman como valores de probabilidad y actividad en las entradas

El error cometido se va a tomar en términos absolutos:

$$e = a^{disj} - a \tag{4.1}$$

Siendo a^{disj} la actividad de conmutación calculada mediante la partición por regiones disjuntas. Mientras que a es la actividad de conmutación calculada sin partición por regiones disjuntas, que bien puede haber sido hallada a través del circuito en puertas o en RTL-exacto, debido a que dan el mismo resultado.

Tomar el error en términos absolutos y no en términos relativos es una práctica habitualmente empleada ([119], [80], [7]) ya que al ser una probabilidad, los errores relativos para valores

pequeños se hacen muy grandes y no tienen una correspondencia real[48]. Y es que hasta las estadísticas de los vectores de test generados aleatoriamente con actividades muy bajas (0,016) pueden dar errores relativos del 25%.

Ya se ha comentado que en esta tesis se considerará como error a aquellos casos cuya diferencia supere 0,01. Pues se ha visto que valores cercanos a 0,01 se dan con el análisis simulativo[49].

Desafortunadamente, **los valores de error obtenidos en esta tesis no son comparables con las propuestas previas** porque no se trabaja en el mismo nivel, y, por lo tanto, los análisis no se pueden realizar sobre los mismos circuitos. Además, por trabajar en RTL, no se dispone de los mismos nodos internos que se tienen en puertas. Habitualmente, el resto de propuestas comparan las actividades de todos los nodos internos del circuito, ya que éstos, en puertas, están perfectamente definidos. Así las estadísticas del error (media, varianza,...) se calculan sobre todo el conjunto de nodos internos. Esto hace que la media del error sea menor, puesto que es de esperar que los nodos cercanos a las entradas tengan menor error que las salidas. En esta tesis, las estadísticas de error se realizan únicamente sobre las salidas del circuito, extrayéndose de los resultados de asignar los 1024 pares de valores de la malla (figura 4-10). Esto tiene dos consecuencias que empeoran las estadísticas del error:

- Es de esperar que a las salidas los errores sean mayores por tener más profundidad, haber más complejidad en la lógica y porque los errores puedan haber aumentado por la propagación de las actividades con error a través del circuito.
- Al asignar todos los valores de la malla (figura 4-10) se analiza el circuito en un gran número de condiciones diferentes. Algunas de estas condiciones son extremas y poco probables de darse en un funcionamiento normal del circuito. En las otras propuestas se asignan unos pocos pares de probabilidad-actividad.

Aún así, para poder tener una valoración del error, en la tabla 4-9 se incluyen datos de los errores cometidos en diversas propuestas. Estas propuestas han realizado los experimentos sobre los circuitos del ISCAS'85 [15].

La segunda columna (*Puertas*) de la tabla 4-9 indica el número de puertas del circuito y la tercera columna (*P. In*) el número de puertos de entrada del circuito. Como se puede comparar con la tabla 4-1, que muestra los circuitos utilizados en esta tesis, excepto los dos últimos circuitos, los circuitos son de tamaño similar. Sin embargo, estos dos últimos circuitos (*c3540* y *c6288*) así como otros ISCAS'85 están descritos sin utilizar puertas XOR y XNOR. Esto hace que el número de puertas aumente considerablemente ya que son circuitos aritméticos y éstos contienen muchas puertas XOR. Y se necesitan tres o cuatro puertas (AND, OR e inversores) para implementar una puerta XOR. El circuito *c3540* es una ALU y el circuito *c6288* es un multiplicador.

Como se puede apreciar en la tabla 4-9, en muchos casos el error máximo supera 0,01, que es el valor a partir del cual en esta tesis se considera error. Estos datos pueden ser contrastados con la tabla 4-28 de la sección 4.4.5, en la que se muestran el resumen de los errores del método propuesto. La tabla AIII-18 del apartado AIII.8 del anexo es una ampliación de la tabla 4-28.

[48] Por ejemplo, si la $P(A='1')_{estim}$=0,9 y $P(A='1')_{exacta}$=0,95; el error relativo sería $e_{rel}(A='1')$=(0,95–0,90)/0,95 = 0,053. Si del mismo ejemplo tomásemos las probabilidades de la señal de ser cero, tendríamos: $P(A='0')_{estim}$=0,1 y $P(A='0')_{exacta}$=0,05; entonces $e_{rel}(A='0')$=(0,05–0,10)/0,05 = 1. Obsérvese la diferencia tratándose del mismo error.

[49] Al analizar las estadísticas de una gran cantidad de valores aleatorios (>10^4) con una probabilidad y actividad determinadas se observa un error de este orden.

Circuito	Puertas	P. In	*Schneider* [119]	*Marculescu* [80]			*Bhanja* [8]		
			μ	μ	σ	e_{max}	μ	σ	e_{max}
c432	160	36	0,016	0,028	0,04	0,21	0,002	0,03	0,197
c499	202	41	-	0,013	0,01	0,062	0	0	0,006
c880	383	60	0,006	0,013	0,02	0,069	0,001	0,01	0,066
c1355	546	41	0,005	0,004	0	0,003	0,001	0,02	0,124
c1908	877	33	0,01	0,009	0,02	0,131	0,001	0,01	0,099
c3540	1669	50	0,014	0,03	0,04	0,201	0,005	0,04	0,252
c6288	2416	32	0,023	0,014	0,02	0,089	0,006	0,02	0,318

Tabla 4-9: Estadísticas de los errores en diversas propuestas

Antes de pasar al análisis de cada circuito se recordarán las condiciones para minimizar el error cometido al realizar la partición (§3.4.3):

1. La señal de selección es independiente de las alternativas
2. La actividad de la señal de selección es baja
3. Existe una baja dependencia temporal en las señales reconvergentes
4. La relación entre fuentes comunes de las alternativas es pequeña respecto a las independientes

La primera condición se ha considerado necesaria para poder realizar la partición. Esta condición y la cuarta son **condiciones estructurales o intrínsecas** del circuito, esto es, su cumplimiento depende de la estructura del circuito. Las condiciones segunda y tercera dependen de los valores de probabilidad y actividad de las señales implicadas, por tanto, son **condiciones circunstanciales o extrínsecas**, ya que su cumplimiento variará según las condiciones a que esté sometido el circuito.

Esto último es importante, y es el motivo por el cual se ha decidido realizar los análisis tomando una malla de valores de probabilidad y actividad. Así se consigue analizar un amplio número de condiciones a las que puede estar sometido el circuito. Esto se verá en el análisis, en el que para determinadas circunstancias el error aumentará o disminuirá según el cumplimiento de las condiciones extrínsecas. Durante el proceso de análisis se puede saber si se cumplen las condiciones extrínsecas, y por tanto, se puede prever la fiabilidad de la estimación y la conveniencia de realizar la partición por regiones disjuntas.

Además, es importante recordar que se deben de realizar el **mínimo** número de particiones disjuntas, **evitando realizar particiones disjuntas exhaustivas**.

A continuación se pasará a analizar el error de cada uno de los circuitos.

4.3.1.1 Circuito "max"

En el apartado 4.2.1 se vio que el ordenamiento RTL para este circuito proporciona un orden muy bueno y se logra que los BDD de actividad de mayor tamaño aumenten linealmente con el número de bits. Este aumento del tamaño de los BDD con el número de bits es bajo y por tanto, realizar partición por regiones disjuntas no ofrece ventajas en cuanto a cálculo respecto al análisis RTL-exacto (éste sí que ofrece ventajas respecto al análisis en puertas).

Si al circuito de 4 bits se le realiza una partición en regiones disjuntas exhaustiva[50] (§3.4.3 y anexo AIII.3.1), de las cuatro señales de salida, hay dos que dan error: $Z(0)$ y $Z(1)$. Mientras que ni $Z(2)$ y ni $Z(3)$ dan error, ya que en ellas no se realiza partición.

Como se verá en el estudio del error de los siguientes circuitos de pruebas, las particiones exhaustivas no son convenientes porque aumentan el error cometido. Es más oportuno realizar

[50] Consúltese la figura AIII-1 del anexo AIII.1 para ver cómo quedaría esta partición para el circuito "max"

el número mínimo de particiones disjuntas que produzcan unos BDD de tamaño mediano/grande. Las particiones por regiones disjuntas permiten limitar el tamaño máximo de los BDD de un diseño. Esto puede ser ventajoso en diseños grandes, pero no en éste porque el ordenamiento RTL produce un aumento controlado del tamaño con el número de bits.

Para el circuito "max", la partición exhaustiva produce BDD de tamaño muy reducido: 6 nodos para los BDD de probabilidad y 20 nodos para los *a*BDD. Más adecuado sería tomar regiones que produzcan *a*BDD con más de un millar de nodos (recuérdese la "alu8051", §4.2.7). Sin embargo, para llegar a ese número, los operandos del circuito tendrían que tener unos 70 bits de ancho (con 20 bits se llega a 290 nodos, §4.2.1).

En consecuencia, para este circuito no es necesario realizar la partición por regiones disjunta, y por ende, **no se produce error en el cálculo**.

Por tanto, sabiendo que en este caso no es conveniente realizar la partición por regiones disjuntas, se ha dejado para el apartado AIII.1.2 del anexo el análisis del error. Este análisis se realizará únicamente con el fin de demostrar las condiciones propuestas en esta tesis para realizar la partición por regiones disjuntas (§3.4.3).

Por último añadir que la decisión de no realizar la partición por regiones disjuntas se puede tomar durante el análisis del circuito, ya que gracias al EHM se dispone de la información sobre los operadores involucrados y la estructura del circuito. En este caso, al existir una operación "mayor que" se puede identificar como que en principio, con este tipo de operador no es preciso realizar la partición.

4.3.1.2 Circuito "comparador"

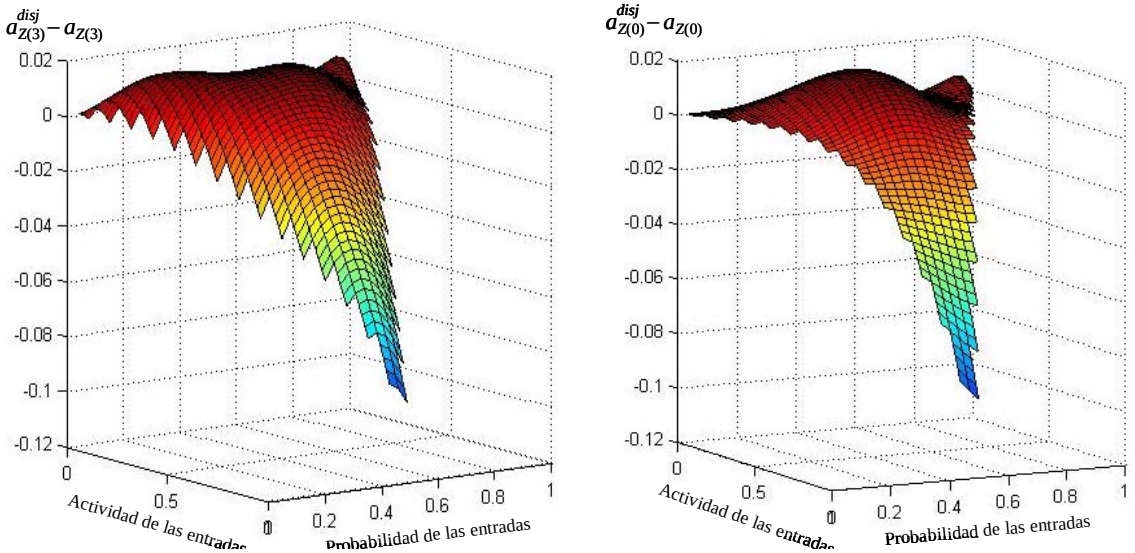
En este circuito, las regiones reconvergentes y las disjuntas coinciden. Por tanto, **en ninguna de las señales se comete error**.

4.3.1.3 Circuito "alu_peq"

Pese al pequeño tamaño de este circuito, la partición por regiones disjuntas no exhaustiva ofrece una interesante reducción del número de nodos (§4.2.3 y anexo AIII.3). Es de esperar que con un mayor número de bits de los operandos, la reducción sea más importante (como ocurre en los circuitos "alu8051_simp" y "alu8051").

Cuando se realiza partición por regiones disjuntas, de las 7 señales de salida de esta ALU (figura 4-3), las señales de resultado (*Z*) dan un error que debe ser estudiado, la señal *Flags*(2) da un error despreciable en todos los casos, y las señales *Flags*(1) y *Flags* (0) no dan error.

La distribución del error de los cuatro bits de la señal *Z* es similar. En la gráfica 4-16 se muestra la distribución del error para las señales *Z*(3) y *Z*(0) para todos los pares de probabilidad-actividad de la malla (recuérdese la figura 4-10).



Gráfica 4-16: Distribución del error absoluto de las señales Z(3) y Z(0) del circuito "alu_peq" (circuito 3) para todo el rango de probabilidades y actividades posibles de las entradas (mismo valor para todas las entradas)

En la tabla 4-10 se muestran los errores medios, las desviaciones típicas y los errores máximos negativo y positivo en los puertos de salida. Para el cálculo de los errores medios, los valores de error se han tomado sin signo.

Señales	μ	σ	e_{max-}	e_{max+}
Z(3)	0,015	0,016	-0,109	+0,016
Z(2)	0,015	0,017	-0,109	+0,016
Z(1)	0,014	0,017	-0,109	+0,019
Z(0)	0,012	0,016	-0,109	+0,016
Flags(2)	0,0001	0,0001	-0,0004	+0,0004
Flags(1)	0	0	0	0
Flags(0)	0	0	0	0

Tabla 4-10: Error medio (tomados en valor absoluto), desviación típica, y errores máximos para cada puerto de salida. Las condiciones en las entradas son las mismas para todos los puertos.

La distribución de los valores absolutos de los errores se muestra en la siguiente tabla (tabla 4-11).

distribución	$\|e\| < 0,01$	$0,01 < \|e\| < 0,02$	$0,02 < \|e\| < 0,03$	$0,03 < \|e\| < 0,04$	$0,04 < \|e\| < 0,05$	$\|e\| > 0,05$
Z(3)	48,4 %	33,9 %	5,1 %	3,7 %	2,9 %	6,0 %
Z(2)	50,0 %	32,6 %	4,8 %	3,8 %	2,8 %	6,0 %
Z(1)	51,2 %	31,9 %	5,1 %	3,3 %	2,6 %	5,9 %
Z(0)	64,4 %	21,5 %	3,9 %	3,2 %	2,1 %	4,9 %
Flags(2)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(1)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(0)	100 %	0 %	0 %	0 %	0 %	0 %

Tabla 4-11: Distribución de los errores. Las condiciones en las entradas son las mismas para todos los puertos.

La señal *Flags* no tiene error y mientras que los bits de la señal *Z* sí tienen. Como se ha dicho, errores menores de 0,01 se consideran despreciables. En la tabla 4-11 se observa que el grupo de los errores menores que 0,01 es el más numeroso, rondando la mitad del total. Si este grupo se une al siguiente, los errores menores de 0,02 agrupan a más del 80% de los casos.

Un análisis detallado muestra que el error máximo positivo no es grande, ya que no supera 0,02, ni tampoco lo es el error medio. La zona en la que los errores son mayores (negativos) es donde las actividades son altas. El error máximo negativo se da para P_{in} = 0,5 y a_{in} = 1. Estas condiciones implican que en cada ciclo del circuito todos los bits de las señales de entrada cambian de valor. Esto es una **situación irreal** para un circuito en funcionamiento normal, y además hace que las entradas se comporten de manera determinista (ya que siempre van a cambiar de valor). Esta situación provoca que no se cumplan las condiciones extrínsecas para la partición en regiones disjuntas (§3.4.3), en concreto:

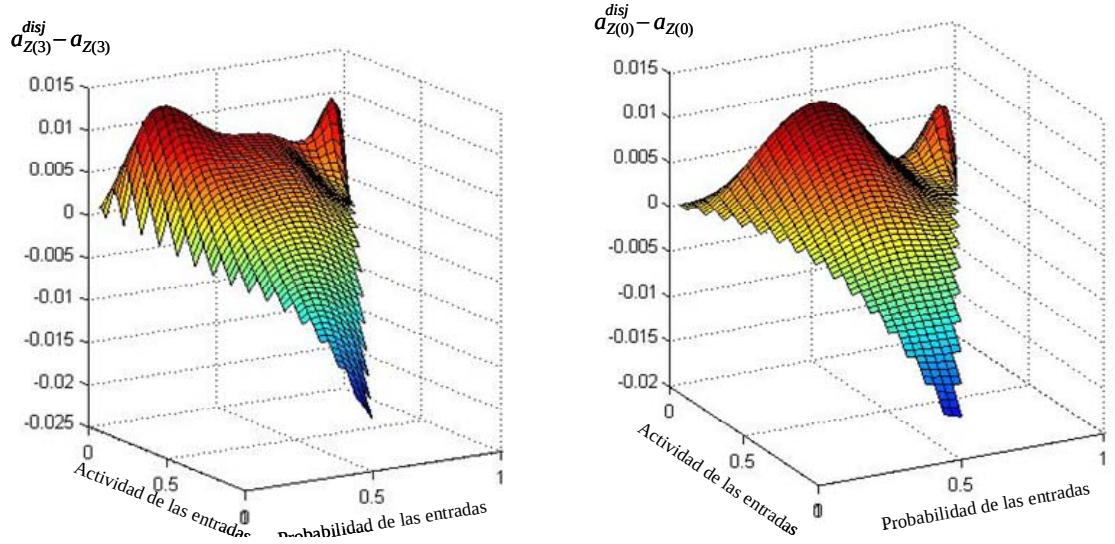
- La condición 2: baja actividad de la señal de selección
- La condición 3: baja correlación temporal de las señales reconvergentes

En el apartado AIII.3 del anexo se realiza un análisis más profundo de las causas del error y de la partición en regiones disjuntas. Ahora, mediante dos experimentos, se va a demostrar la influencia de estas dos condiciones por separado.

Un primer experimento **limita la actividad de las señales de selección** a 0,1. El resto de señales cubren todo el rango de probabilidades y actividades. Por lo tanto, en un caso, los bits de la señal *Z* mantendrán condiciones de entrada irreales (P_{in} = 0,5 y a_{in} = 1), sin embargo, la señal de selección (*Code*) tendrá unos valores de actividad reales.

Limitar la actividad de conmutación de los bits de la señal de selección a 0,1 implica que la señal de selección, considerada de forma conjunta y no bit a bit, tenga una actividad a_{code} = 0,27. Por tanto, de media, ésta cambiará de valor una vez cada 3 ó 4 ciclos[51], lo cual puede ser un valor razonable.

En la gráfica 4-17 se muestra la distribución del error de las señales *Z*(3) y *Z*(0) con la limitación de la actividad de conmutación de la señal *Code*. Compárense estas gráficas con las que no tienen la limitación de la actividad de conmutación (gráfica 4-16).



Gráfica 4-17: Distribución del error absoluto de Z(3) y Z(0) del circuito "alu_peq" (circuito 3) para todo el rango de probabilidades y actividades, limitando la actividad de las señales de selección a 0,1

Los errores medios, desviación típica y errores máximos se muestran a continuación (tabla 4-12)

[51] La actividad de la señal *code* considerada de forma conjunta (y no bit a bit) se calcula con la fórmula:
$a_{code} = a_{code(0)} + a_{code(1)} + a_{code(2)} - a_{code(0)} \cdot a_{code(1)} - a_{code(0)} \cdot a_{code(2)} - a_{code(1)} \cdot a_{code(2)} + a_{code(0)} \cdot a_{code(1)} \cdot a_{code(2)}$

Señales	μ	σ	e_{max-}	e_{max+}
Z(3)	0,0051	0,0037	-0,021	+0,012
Z(2)	0,0050	0,0037	-0,021	+0,013
Z(1)	0,0049	0,0039	-0,020	+0,015
Z(0)	0,0037	0,0033	-0,019	+0,010
Flags(2)	0,0001	0,0001	-0,0004	+0,0004
Flags(1)	0	0	0	0
Flags(0)	0	0	0	0

Tabla 4-12: Error medio (tomados en valor absoluto), desviación típica, y errores máximos para cada puerto de salida. Se ha limitado la actividad de las señales de selección a 0,1.

La distribución de los valores absolutos de los errores se muestra en la tabla 4-13.

distribución	$\|e\| < 0,01$	$0,01 < \|e\| < 0,02$	$0,02 < \|e\| < 0,03$	$0,03 < \|e\| < 0,04$	$0,04 < \|e\| < 0,05$	$\|e\| > 0,05$
Z(3)	90,3 %	9,5 %	0,2 %	0 %	0 %	0 %
Z(2)	89,9 %	10 %	0,1 %	0 %	0 %	0 %
Z(1)	87,5 %	12,4 %	0,1 %	0 %	0 %	0 %
Z(0)	95,6 %	4,4 %	0 %	0 %	0 %	0 %
Flags(2)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(1)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(0)	100 %	0 %	0 %	0 %	0 %	0 %

Tabla 4-13: Distribución de los errores limitando la actividad de las señales de selección a 0,1

La comparación de las gráficas y tablas de este experimento con el experimento inicial confirma la influencia de la actividad de la señal de selección en el error. Por tanto, como la actividad de la señal de selección es un dato conocido a la hora de realizar la estimación[52], **se podrá valorar la influencia de esta actividad en el error** y **decidir si conviene realizar la partición**.

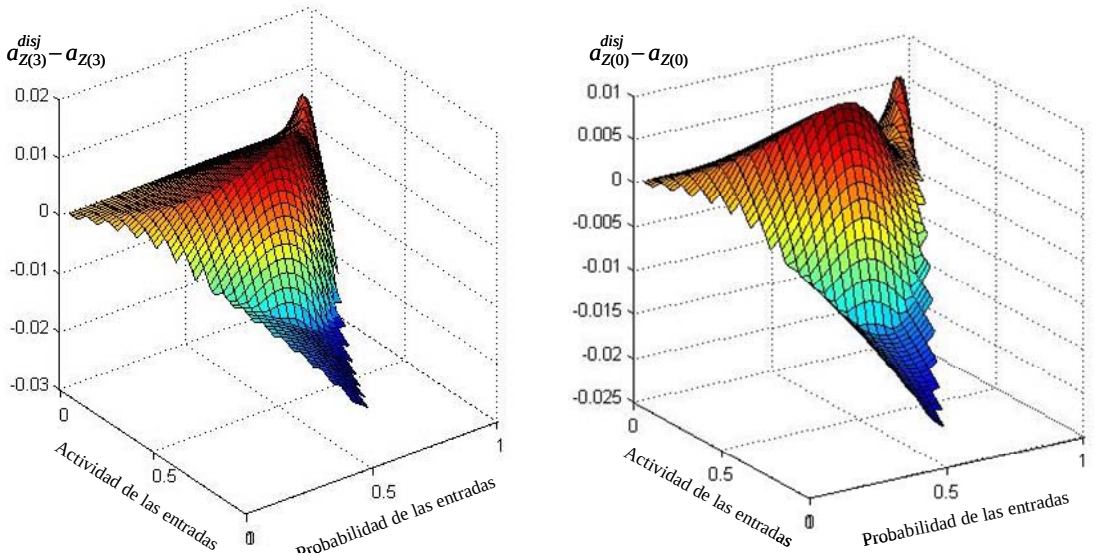
De las tablas y las gráficas se puede observar que el máximo error se sigue produciendo en el caso poco probable de $P_{in} = 0,5$ y $a_{A(i)} = a_{B(i)} = 1$; aunque la actividad $a_{Code(i)} = 0,1$.

El segundo experimento **limita la correlación temporal de las señales reconvergentes**. Los puertos de entrada *A* y *B* tendrán limitada su actividad de modo que:

- Cuando $a - a^{ti} > 0,1$ se asignará una actividad $a = a^{ti} + 0,1$
- Cuando $a^{ti} - a > 0,1$ se asignará una actividad $a = a^{ti} - 0,1$

Para las señales de selección (*Code*) se asigna la actividad que le corresponda, incluido el caso extremo $P_{in} = 0,5$ y $a_{in} = 1$. La distribución del error de las señales *Z*(3) y *Z*(0) de este experimento se muestra en la gráfica 4-18.

[52] El proceso de estimación se realiza desde las entradas a las salidas, por tanto, las actividades de las dependencias, lo que incluye a la señal de selección, se han estimado previamente.



Gráfica 4-18: Distribución del error absoluto de Z(3) y Z(0) del circuito "alu_peq" (circuito 3) para todo el rango de probabilidades y actividades, limitando la correlación temporal de los operandos A y B.

Los errores medios, desviación típica y errores máximos se muestran a continuación (tabla 4-14)

Señales	μ	σ	e_{max-}	e_{max+}
Z(3)	0,0065	0,0058	-0,022	+0,011
Z(2)	0,0065	0,0058	-0,022	+0,011
Z(1)	0,0064	0,0058	-0,022	+0,011
Z(0)	0,0057	0,0053	-0,022	+0,010
Flags(2)	0	0	0	0
Flags(1)	0	0	0	0
Flags(0)	0	0	0	0

Tabla 4-14: Error medio (tomados en valor absoluto), desviación típica, y errores máximos para cada puerto de salida. Se ha limitado la correlación temporal de los operandos A y B.

La distribución de los valores absolutos de los errores se muestra en la tabla 4-15.

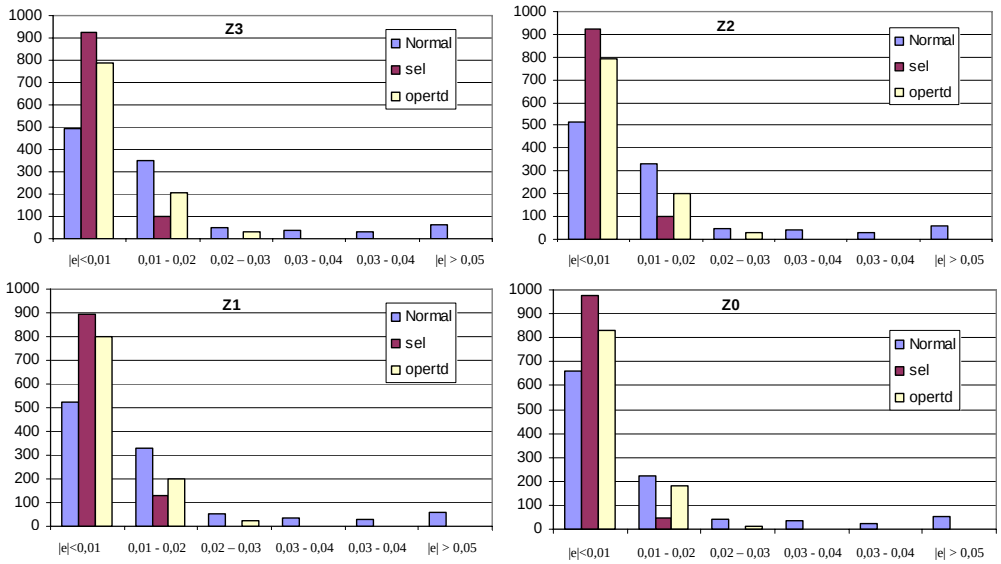
distribución	$\|e\| < 0,01$	$0,01 < \|e\| < 0,02$	$0,02 < \|e\| < 0,03$	$0,03 < \|e\| < 0,04$	$0,04 < \|e\| < 0,05$	$\|e\| > 0,05$
Z(3)	77,2 %	19,9 %	2,9 %	0 %	0 %	0 %
Z(2)	77,7 %	19,6 %	2,7 %	0 %	0 %	0 %
Z(1)	78,1 %	19,5 %	2,4 %	0 %	0 %	0 %
Z(0)	81,0 %	18,0 %	1,0 %	0 %	0 %	0 %
Flags(2)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(1)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(0)	100 %	0 %	0 %	0 %	0 %	0 %

Tabla 4-15: Distribución de los errores limitando la correlación temporal de los operandos A y B

Estos resultados también mejoran los resultados iniciales. Se debe tener en cuenta que en este experimento la actividad de la señal de selección no se ha limitado, y que los peores casos se siguen correspondiendo con la situación extrema en la que: P_{in} = 0,5 y $a_{Code(i)}$ = 1; aunque la actividad de los operandos sea $a_{A(i)} = a_{B(i)}$ = 0,6.

En la gráfica 4-19 se compara la distribución de los errores para los bits de la señal *Z* en las tres condiciones que se han realizado los experimentos. La primera barra se corresponde con el experimento normal, en donde todas las señales de entrada toman todos los posibles pares de probabilidad y actividad. La segunda barra se refiere al experimento que limitaba la actividad

de la señal de selección (*Code*). La tercera barra se corresponde con el experimento que limita la dependencia temporal de las señales reconvergentes: los operandos *A* y *B*.



Gráfica 4-19: Comparación de la distribución de los errores en los tres experimentos: 1) normal, 2) con limitación en la actividad de la señal de selección (sel), 3) con limitación en la dependencia temporal de los operandos (opertd)

Estos dos experimentos que limitan la actividad de la señal de selección o la dependencia temporal de los operandos se han aplicado de forma separada. En ambos casos, los errores máximos se siguen dando en las condiciones extremas de P_{in} = 0,5 y a_{in} = 1 para las señales que no se han limitado. Consecuentemente, en los casos en los que se apliquen ambos límites simultáneamente, el error será más reducido aún.

De los análisis realizados para este circuito se puede concluir que:

- Realizando una partición en regiones disjuntas no exhaustiva del circuito (ver apéndice AIII.3) **en la mayoría de los casos se obtiene un error** que se puede considerar **aceptable** (menor que 0,02). Además, como se vio en el apartado §4.2.3, esta partición proporciona unos buenos tamaños de los BDD.
- Hay unos pocos casos en los que el error es grande. Sin embargo estos casos **se dan en condiciones muy poco probables** y que **pueden ser detectadas en el momento de la estimación**. Por tanto, la magnitud del error puede ser valorada, y así decidir si conviene realizar la partición por regiones disjuntas.
- Estas condiciones que pueden hacer que el error sea elevado hace que no se cumplan las condiciones 2 y 3 propuestas para la partición por regiones disjuntas (§3.4.3), y son:
    - Que la actividad de la actividad de selección no sea muy elevada
    - Que la correlación temporal de las señales reconvergentes no sea muy grande

Por tanto, de todo lo visto en este apartado, se puede afirmar que para este circuito, la partición disjunta mínima proporciona una baja probabilidad de error junto con un buen tamaño de BDD. Si bien, por el tamaño del circuito, no sea necesario recurrir a la partición disjunta.

4.3.1.4 Circuito "alu_core"

En el apartado 4.2.4 se vio que la partición por regiones disjuntas proporciona tamaños ligeramente inferiores al análisis en RTL sin partición por regiones disjuntas. Por tanto, ya que no supone mucha ventaja, no es necesario realizar la partición. Además, como el la mayoría de

las operaciones del circuito son a nivel de bits, las partición por regiones disjuntas no proporciona una ventaja significativa al aumentar el ancho de bus de los operandos.

Sin embargo, **la partición disjunta mínima no produce error**. En todos los casos, el error es más bajo que 0,0001, lo que es bastante más bajo de lo que se ha considerado como despreciable en esta tesis (0,01).

A título descriptivo, en el apartado AIII.4 del anexo se analiza el error de la partición disjunta exhaustiva. Allí se verá que este error es poco importante, sólo en ciertos casos extremos se da un error a tener en cuenta.

4.3.1.5 Circuito "addsub"

En el apartado 4.2.5 se ha visto que la partición por regiones disjuntas ofrece importantes beneficios en el uso de recursos. En este apartado se analizarán las condiciones que hacen que el error sea aceptable.

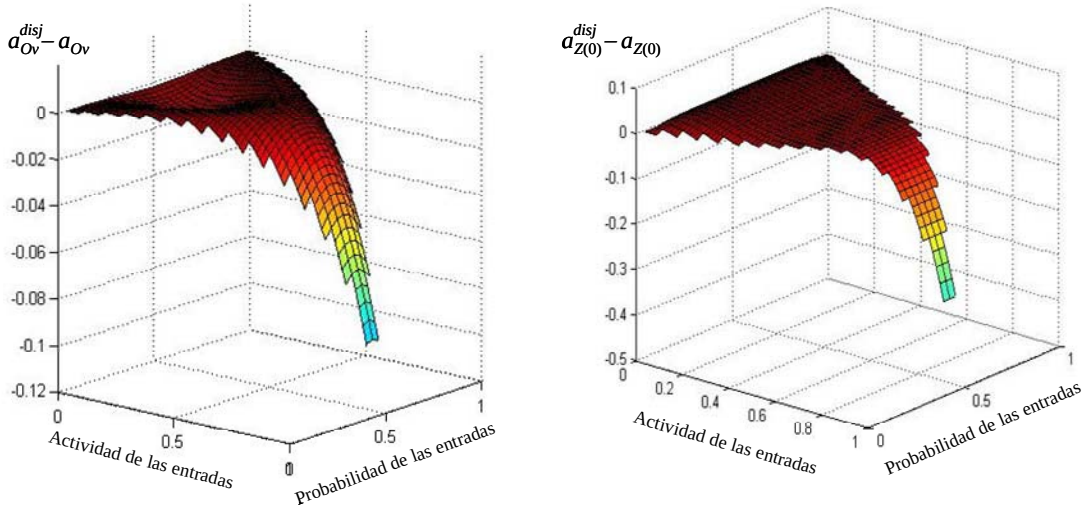
Se ha realizando el mismo análisis que se ha venido haciendo en los circuitos anteriores, en el que todas las entradas van tomando el conjunto de valores posibles de probabilidad y actividad (figura 4-10). Para este análisis, el error medio, la desviación típica y los errores máximos para cada señal se muestran en la tabla 4-16.

Señales	μ	σ	e_{max-}	e_{max+}
Z(7)	0,0003	0,0005	-0,0002	0,002
Z(6)	0,0004	0,0009	-0,005	0,003
Z(5)	0,0005	0,0010	-0,005	0,003
Z(4)	0,0007	0,0010	-0,002	0,004
Z(3)	0,0010	0,0014	-0,003	0,006
Z(2)	0,0017	0,0019	-0,004	0,008
Z(1)	0,0042	0,0043	-0,020	0,011
Z(0)	0,0206	0,0442	-0,50	0,026
Co(1)	0,00005	0,0001	-0,003	0,0001
Co(0)	0,0004	0,0006	-0,004	0,0006
Ov	0,0072	0,0146	-0,125	0,0009

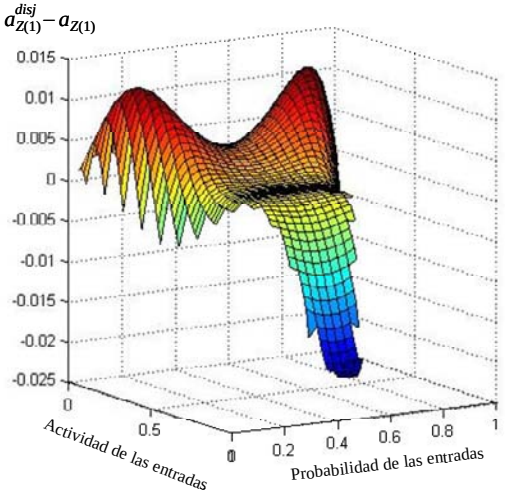
Tabla 4-16: Errores máximos para cada puerto de salida y las condiciones de probabilidad y actividad a las entradas que los producen. Las condiciones en las entradas son las mismas para todos los puertos.

De la tabla se observa que sólo son significativos los errores en *Z*(1), *Z*(0) y *Ov*. Para el resto de señales los errores están entre -0,01 y 0,01.

Para *Z*(0) el máximo error negativo es muy elevado (-0,5). Sin embargo, tal como ha ocurrido en otras ocasiones, este error se da en condiciones extremas, cuando P_{in} = 0,5 ; a_{in} = 1. En las gráficas 4-20 y 4-21 se han representado las distribuciones de los errores para las señales con mayor error (*Ov*, *Z*(0), *Z*(1)). En ellas se aprecia que es en la zona de máxima actividad donde se dan estos errores máximos negativos.



Gráfica 4-20: Distribución de los errores absolutos de Ov y Z(0) del circuito 5 para todo el rango de probabilidades y actividades



Gráfica 4-21: Distribución del error absoluto de Z(1) del circuito 5 para todo el rango de probabilidades y actividades

La distribución de los errores para todas las señales se muestra en la tabla 4-17, en donde se ve que sólo son significativos para *Z*(0) y *Ov*.

distribución	$\|e\| < 0{,}01$	$0{,}01 < \|e\| < 0{,}02$	$0{,}02 < \|e\| < 0{,}03$	$0{,}03 < \|e\| < 0{,}04$	$0{,}04 < \|e\| < 0{,}05$	$\|e\| > 0{,}05$
Z(7)	100 %	0 %	0 %	0 %	0 %	0 %
Z(6)	100 %	0 %	0 %	0 %	0 %	0 %
Z(5)	100 %	0 %	0 %	0 %	0 %	0 %
Z(4)	100 %	0 %	0 %	0 %	0 %	0 %
Z(3)	100 %	0 %	0 %	0 %	0 %	0 %
Z(2)	100 %	0 %	0 %	0 %	0 %	0 %
Z(1)	90,3 %	8,8 %	0,9 %	0 %	0 %	0 %
Z(0)	48,1 %	22,7 %	20,3 %	1,8 %	0,9 %	6,2 %
Co(1)	100 %	0 %	0 %	0 %	0 %	0 %
Co(0)	100 %	0 %	0 %	0 %	0 %	0 %
Ov	85,4 %	5,5 %	2,6 %	2,2 %	1,4 %	2,9 %

Tabla 4-17: Distribución de los errores para el circuito "addsub"

SIn embargo, el **error de la señal *Z*(0) es muy fácil de evitar** ya que *Z*(0) sólo depende de 4 puertos de entrada: *A*(0), *B*(0), *Addsub*, *Ci*. Por tanto, para el cálculo de *Z*(0) **no se debería de realizar partición**, ya que sus BDD no serán muy grandes. La decisión de no realizar la partición para *Z*(0) se toma durante el proceso de la partición, ya que **el modelo extendido del hardware** (EHM, §3.3.2) **permite conocer de manera inmediata el número de dependencias**. Esto va en consonancia con evitar realizar particiones disjuntas exhaustivas (§3.4.3). Con esto se consigue **eliminar totalmente su error** manteniendo tamaños de BDD similares. Los tamaños de los BDD resultantes de los dos tipos de partición (partiendo o no en la señal *Z*(0)) se muestran en la tabla 4-18.

ancho de bus: 8 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
RTL-disjunto	45	187	645	375	5	19	13
RTL-disjunto-sin *Z*(0)	43	179	621	363	5	19	13

Tabla 4-18: Comparación de los tamaños de los BDD sin dividir la señal Z(0) en regiones disjuntas

Observando la tabla 4-18 se aprecia que al no dividir las regiones disjuntas de *Z*(0), no sólo se consiguen unos tamaños de BDD similares, sino que se logra una ligera disminución.

Por tanto, como ya se ha comentado, **conviene realizar la partición por regiones disjuntas en regiones de tamaños medios**, evitando hacerlas lo más pequeñas posible. Con esto se consigue disminuir el error cometido a la vez que se mantienen los requerimientos de cómputo. Esto es parte de la estrategia de realizar particiones disjuntas mínimas, en vez de particiones disjuntas exhaustivas.

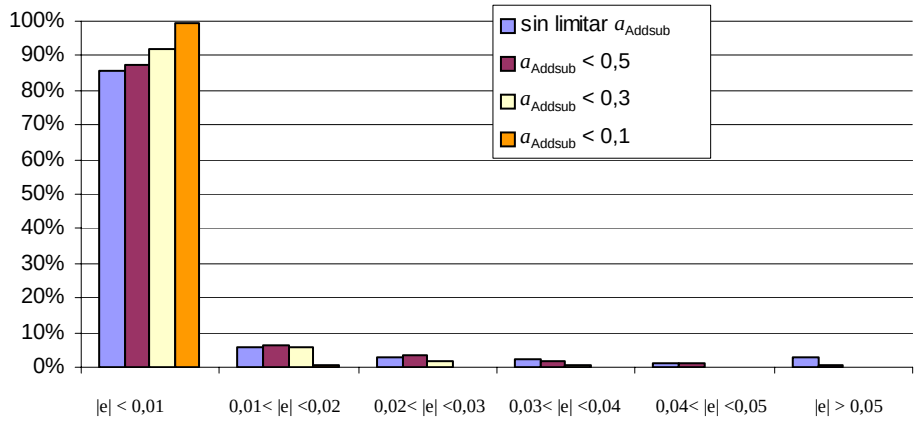
De la misma manera, si se quiere proceder a **eliminar el error de** *Z*(1) se puede no realizar la partición para esta señal, ya que sólo depende de seis señales.

El **error de la señal *Ov*** está restringido a las zonas de mucha actividad, para el resto de las zonas es pequeño (el 85,4% de los casos $|e| < 0{,}01$ y el 90,9% $|e| < 0{,}02$). Para las zonas de mucha actividad el error negativo aumenta (ver gráfica 4-20). Este es el mismo problema que se vio en el circuito "alu_peq" (circuito 3), ya que las situaciones en las que se da este error son irreales ya que la actividad es muy alta (véase el apartado AIII.3.1 del anexo). En estos casos, la señal de selección *Addsub* tiene una actividad muy alta y hace que aumente el error. Si se repite el experimento limitando la actividad de *Addsub* lo errores son menores. En la tabla 4-19 se muestra la media el error, la desviación típica y los errores máximos para distintos experimentos en los que se ha limitado la actividad de *Addsub*. También se incluyen las condiciones en las que se dan los errores máximos, que se dan cuando $P_{in} = 0{,}5$; $a_{in} = 1$

Error de *Ov*	μ	σ	Error absoluto máximo negativo (e_{max-})			Error absoluto máximo positivo (e_{max+})		
			P_{in}	a_{in}	e_{max-}	P_{in}	a_{in}	e_{max+}
sin limitar a_{Addsub}	0,0072	0,0146	0,5	1	-0,125	0,781	0,266	0,0009
a_{Addsub}< 0,5	0,0053	0,0084	0,5	1	-0,062	0,781	0,266	0,0009
a_{Addsub}< 0,3	0,0038	0,0052	0,5	1	-0,037	0,781	0,266	0,0009
a_{Addsub}< 0,1	0,0018	0,0021	0,5	1	-0,012	0,187	0,219	0,0004

Tabla 4-19: Error medio, desviación típica y errores máximos para Ov. Se incluyen las condiciones donde se dan los errores máximos. Las condiciones en las entradas son las mismas para todos los puertos excepto para a_{Addsub}

La distribución de los errores de la señal *Ov* también mejora cuanto más se limite la actividad de *Addsub*. En la gráfica 4-22 se muestra la distribución del error según se limite la actividad de *Addsub*.



Gráfica 4-22: Distribución de los errores de Ov según el valor máximo de actividad de Addsub

Como se ve en la gráfica 4-22, en todos los casos la mayoría de los errores son menores de 0,01. Y cuando $a_{Addsub} \leq 0{,}1$ se puede considerar que casi no hay error. La tabla 4-20 muestra los valores numéricos de la gráfica 4-22.

	$\|e\| < 0{,}01$	$0{,}01 < \|e\| < 0{,}02$	$0{,}02 < \|e\| < 0{,}03$	$0{,}03 < \|e\| < 0{,}04$	$0{,}04 < \|e\| < 0{,}05$	$\|e\| > 0{,}05$
sin limitar a_{Addsub}	85,4 %	5,5 %	2,6 %	2,2 %	1,4 %	2,9 %
a_{Addsub}< 0,5	87,2 %	6,5 %	3,3 %	1,7 %	0,9 %	0,4 %
a_{Addsub}< 0,3	92,0 %	5,8 %	1,8 %	0,4 %	0 %	0 %
a_{Addsub}< 0,1	99,5 %	0,5 %	0 %	0 %	0 %	0 %

Tabla 4-20: Distribución de los errores de Ov. Las condiciones en las entradas son las mismas para todos los puertos excepto para a_{Addsub}

Para el resto de señales los errores y su distribución los resultados también son más favorables limitando la actividad de *Addsub*. No se ha mostrado su análisis porque ya sin limitar a_{Addsub} tenían errores suficientemente bajos (menores que 0,01).

Para el circuito con un **mayor número de bits** de los operandos los resultados son similares. El error en los 8 primeros bits del resultado (*Z*) de un sumador-restador de más de 8 bits es el mismo que para el sumador/restador de 8 bits. En los siguientes bits el error disminuye. Así, en un sumador/restador de 12 bits, todos los errores de los bits 8 a 11 están entre -0,01 y 0,01. Lo mismo ocurre con los acarreos (*Co*).

El error de la señal de desbordamiento es prácticamente el mismo para 12 bits que para 8 bits. Para 12 bits, la señal *Ov* tiene los mismos errores máximos que en los que figura en la tabla 4-16, y ligeramente mejor para el de 12 bits, siendo prácticamente la misma distribución de error que la de la tabla 4-17.

Como conclusión, se ha visto que realizando la partición adecuada, evitando partir para *Z*(0) y *Z*(1), **ninguna señal tiene error salvo** *Ov*. La señal *Ov* sólo tiene error cuando se dan condiciones de muy alta actividad. Estas condiciones no son habituales y pueden ser detectadas a la hora de realizar la partición.

4.3.1.6 Circuito "alu8051_simp"

Este circuito no permite realizar partición por regiones reconvergentes (§4.2.6), por tanto, para limitar de manera efectiva el tamaño de los BDD se necesita realizar partición por regiones disjuntas. Realizando esta partición el tamaño máximo de los BDD queda limitado a un valor fijo, independientemente del número de bits de los operandos (ver tabla AIII-16 del apartado AIII.6 del anexo). Consecuentemente, con el número de bits se tiene un aumento lineal del número total de nodos, y se logra acotar el problema. Como contrapartida, la partición por regiones disjuntas hace que se cometa un error en el cálculo.

Sin embargo, el error cometido para este circuito es pequeño. La tabla 4-21 muestra la distribución del error para cada uno de los puertos de salida. Nótese que hay muy pocos errores mayores que 0,01; y no hay errores mayores que 0,02.

distribución	$\|e\| < 0{,}01$	$0{,}01 < \|e\| < 0{,}02$	$\|e\| > 0{,}02$
Z(7)	97,9 %	2,1 %	0 %
Z(6)	97,8 %	2,2 %	0 %
Z(5)	97,8 %	2,2 %	0 %
Z(4)	98,0 %	2,0 %	0 %
Z(3)	98,1 %	1,9 %	0 %
Z(2)	98,8 %	1,2 %	0 %
Z(1)	100 %	0 %	0 %
Z(0)	98,7 %	1,3 %	0 %
Co(1)	100 %	0 %	0 %
Co(0)	100 %	0 %	0 %
Ov	100 %	0 %	0 %

Tabla 4-21: Distribución de los errores en el circuito "alu8051_simp"

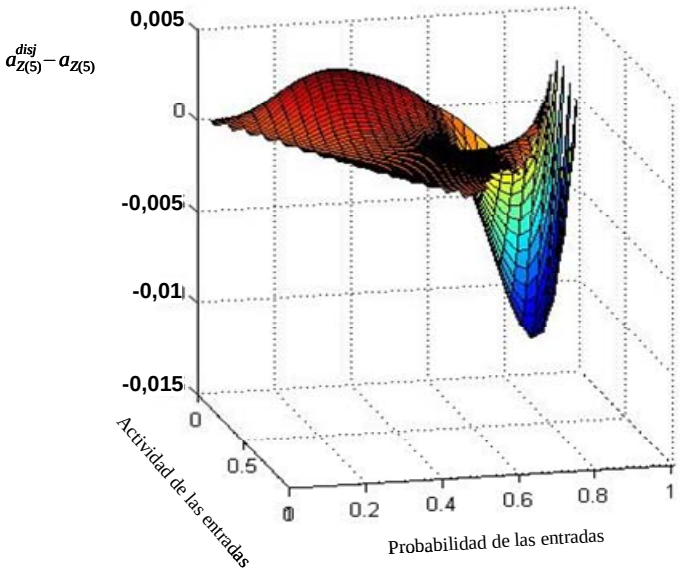
De todas las señales el error máximo negativo se produce en *Z*(5), que es $e_{max-} = -0{,}012$. Y el máximo error positivo se da para *Z*(0), y es $e_{max+} = 0{,}011$. Estos errores son bajos, ya que están muy cerca del valor a partir del cual se ha considerado que se empieza a cometer error: 0,01.

Los errores medios son muy bajos, estando bastante por debajo de 0,01. En la tabla 4-22 se muestran los errores medios (tomados en valor absoluto), desviación típica y errores máximos.

Señales	μ	σ	e_{max-}	e_{max+}
Z(7)	0,0014	0,0023	-0,012	0,003
Z(6)	0,0014	0,0024	-0,012	0,003
Z(5)	0,0014	0,0024	-0,012	0,003
Z(4)	0,0014	0,0023	-0,012	0,003
Z(3)	0,0014	0,0023	-0,012	0,003
Z(2)	0,0014	0,0021	-0,011	0,003
Z(1)	0,0011	0,0015	-0,009	0,003
Z(0)	0,0018	0,0022	-0,005	0,011
Co(1)	0,0002	0,0003	-0,0001	0,0014
Co(0)	0,0003	0,0005	-0,0003	0,0019
Ov	0,0003	0,0004	-0,0004	0,0015

Tabla 4-22: Error medio (tomados en valor absoluto), desviación típica, y errores máximos para cada puerto de salida. Las condiciones en las entradas son las mismas para todos los puertos.

En la gráfica 4-23 se muestra la distribución del error para *Z*(5) para todo el rango de de valores de probabilidad y actividad en las entradas. Como se ve de la gráfica y de la tabla 4-21, hay muy pocos casos (2,2%) que tienen un error superior a 0,01, que es lo que en esta tesis se ha considerado despreciable.



Gráfica 4-23: Distribución del error absoluto de Z(5) para todo el rango de probabilidades y actividades posibles de las entradas (mismo valor para todas las entradas)

Por tanto, para este circuito **el error se puede considerar despreciable**. Una de las causas por la que el error es pequeño se debe a que las particiones son de tamaño mediano (ver el capítulo 4.2.6 y las tablas del anexo AIII.6).

Para este circuito se ha añadido un **nuevo análisis** que tiene en cuenta las probabilidades y actividades considerando los operandos: *A* y *B*. Para ello, en vez de asignar a los operandos todo el rango de probabilidades y actividades, se asigna sólo a uno de los operandos (*B*) y se mantiene constantes las probabilidades y actividades del otro operando (*A*).

Por tanto, en estos experimentos, el puerto *A* tiene unas probabilidades y actividades fijas, y el resto de señales (*B*, *Cmd*, *Ov*, *Ci*) van tomando todo el rango de valores.

En un **primer experimento** al puerto *A* se le asignó $P_{A(i)}$ = 0,5 y $a_{A(i)}$ = 0,5. Por tanto, en este experimento la señal *A* no tiene correlación temporal (§2.2.3.2). Al resto de puertos de entrada se le asignó todo el conjunto de probabilidades y actividades, como se ha venido haciendo hasta ahora. Para estas condiciones, para todos los puertos de salida y en los 1024 casos **el error absoluto no supera 0,01**.

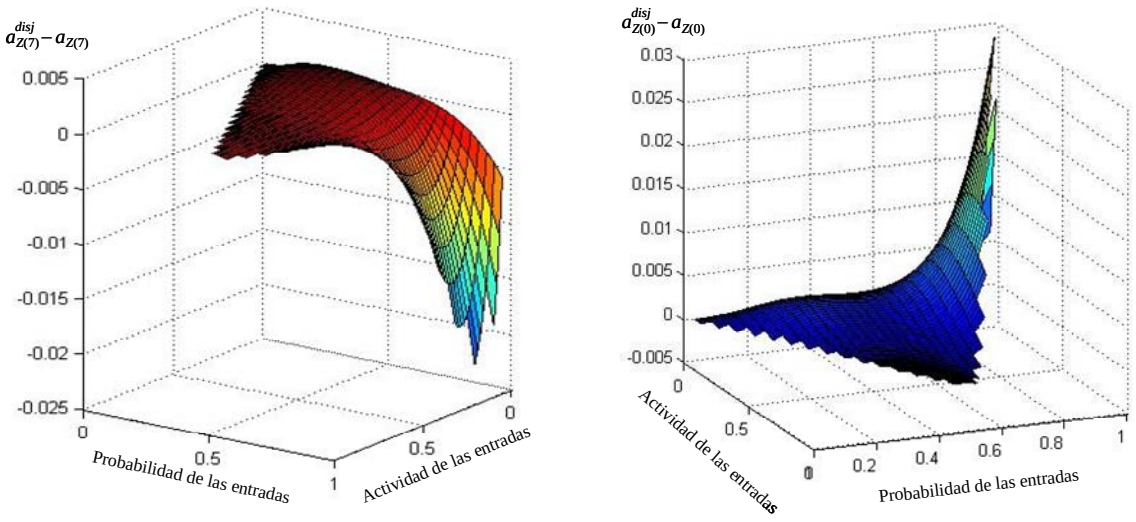
De todas las señales el máximo error positivo se produce en *Co*(1), que es e_{max+} = 0,0014 (para P_i=0,43 ; a_i=0,19) y el máximo negativo es e_{max-} = –0,0010 (para P_i=0,56 ; a_i=0,87). Para el resto de señales menos para *Co*(0) el error está por debajo de 0,0005. En consecuencia, **para este experimento el error se puede considerar que no hay error**.

En el experimento anterior las condiciones del puerto *A* no tienen correlación temporal. Así, se ha realizado un **segundo experimento** en el que se asigna $P_{A(i)}$ = 0,5 y $a_{A(i)}$ = 0,25. En este experimento ya hay un pequeño error debido a la correlación temporal de *A*. Aún así, este error no es importante y se da sólo en ciertos casos. En la tabla 4-23 la distribución del error.

distribución	$\|e\| < 0{,}01$	$0{,}01 < \|e\| < 0{,}02$	$\|e\| > 0{,}02$
Z(7)	96,1 %	3,7 %	0,2 %
Z(6)	96,1 %	3,7 %	0,2 %
Z(5)	96,2 %	3,6 %	0,2 %
Z(4)	96,8 %	3,2 %	0 %
Z(3)	97,8 %	2,2 %	0 %
Z(2)	99,8 %	0,2 %	0 %
Z(1)	100 %	0 %	0 %
Z(0)	93,5 %	4,7 %	1,8 %
Co(1)	100 %	0 %	0 %
Co(0)	100 %	0 %	0 %
Ov	100 %	0 %	0 %

Tabla 4-23: Distribución de los errores manteniendo la probabilidad y actividad de A fija en A en $P_{A(i)}=0{,}5$ y $a_{A(i)}=0{,}25$

En la mayoría de los casos el error es despreciable. Los peores casos se dan en *Z*(7) y *Z*(0), donde en un 3,9% y 6,5% de los casos respectivamente el error absoluto supera 0,01. La gráfica 4-24 muestra la distribución de los errores para estas señales. De estas gráficas se observa que los errores máximos se dan para probabilidades muy altas de entrada $P_i = 0{,}94$. Para probabilidades de entrada menores que 0,82 para *A*(7) y menores que 0,79 para *A*(0) los errores absolutos son menores que 0,01, que es lo que se ha considerado que es despreciable.



Gráfica 4-24: Distribución del error absoluto de Z(7) y Z(0) para todo el rango de probabilidades y actividades posibles de las entradas, excepto para A que se mantienen fijas en $P_{A(i)}=0{,}5$ y $a_{A(i)}=0{,}25$

Los errores medios (tomados en valor absoluto), las desviaciones típicas y los errores máximos negativos y positivos se muestran en la tabla 4-24. Obsérvese que todos los errores medios son menores que 0,01.

Señales	μ	σ	e_{max-}	e_{max+}
Z(7)	0,0017	0,0031	-0,022	0,002
Z(6)	0,0017	0,0031	-0,022	0,002
Z(5)	0,0017	0,0030	-0,021	0,002
Z(4)	0,0016	0,0027	-0,019	0,002
Z(3)	0,0014	0,0023	-0,016	0,002
Z(2)	0,0011	0,0014	-0,010	0,002
Z(1)	0,0007	0,0006	-0,001	0,003
Z(0)	0,0027	0,0044	-0,002	0,029
Co(1)	0,0006	0,0005	-0,002	0,002
Co(0)	0,0003	0,0003	-0,0003	0,001
Ov	0,0002	0,0002	-0,0001	0,001

Tabla 4-24: Error medio (tomados en valor absoluto), desviación típica, y errores máximos para cada puerto de salida. La probabilidad y actividad de A se mantiene fija en A en $P_{A(i)}$=0,5 y $a_{A(i)}$=0,25.

Los resultados de este análisis se pueden considerar válidos, únicamente hay que tener cierto precaución cuando las correlaciones temporales de un operando sean importantes y las probabilidades del otro operando sean elevadas.

Como conclusión, estos resultados permiten validar las estimaciones resultantes de la partición disjunta, incluso realizando variaciones de los valores de probabilidad y actividad agrupados por operandos.

4.3.1.7 Circuito "alu8051"

En el apartado 4.2.7 se mostraron los tamaños de los BDD para dos análisis RTL que realizan particiones en regiones disjuntas de distinto tamaño. El estudio comparado del error en estos dos análisis confirmará que la partición por regiones disjuntas no se debe realizar de forma exhaustiva. Con esto se consigue minimizar el error y permite además un **aumento lineal** del tamaño de los BDD con el ancho de bus.

En la primera partición (RTL-disjunto 1) es la partición disjunta exhaustiva, para el circuito de 8 bits se obtienen 596 regiones cuyos mayores BDD de probabilidad son de tan sólo 5 nodos y 13 nodos para los *a*BDD. Es evidente que con los métodos actuales de cálculo no es preciso llegar a tales extremos y no hay problema por emplear áreas sensiblemente mayores. Por ejemplo, la segunda partición RTL disjunta divide el circuito en 289 regiones, siendo los BDD de probabilidad de 57 como máximo y 912 nodos para los *a*BDD. Estos tamaños son todavía manejables, y la diferencia computacional entre ambos no es significativa.

Una de las principales ventajas de esta **partición en regiones disjuntas** es que en muchos casos **el tamaño máximo de los BDD permanece constante** con el número de bits de los operandos, y por tanto, **el aumento del número de nodos es lineal con el número de bits**. Esto no ocurre con cuando no se realiza la partición disjunta.

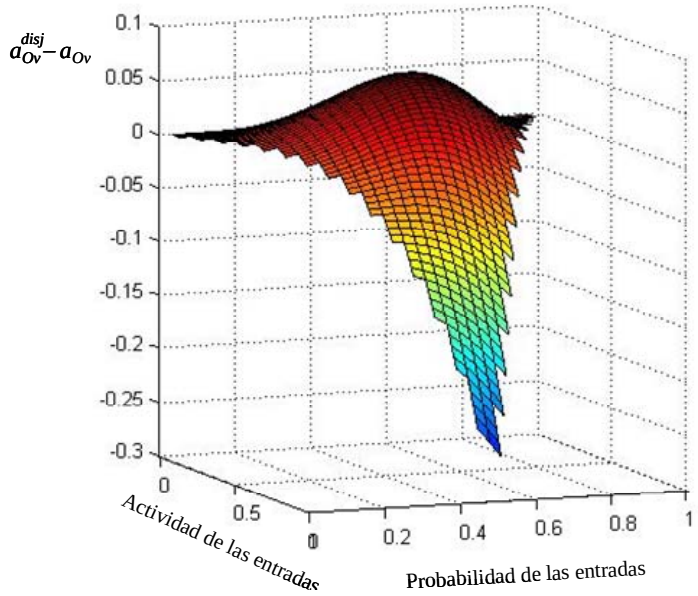
En la tabla 4-25 se muestra la distribución de los errores de todas las salidas para la **partición disjunta exhaustiva (RTL-disjunto 1)**, que es la de máximo número de particiones.

señales	$\|e\| < 0{,}01$	$0{,}01 < \|e\| < 0{,}02$	$0{,}02 < \|e\| < 0{,}03$	$0{,}03 < \|e\| < 0{,}04$	$0{,}04 < \|e\| < 0{,}05$	$\|e\| > 0{,}06$
Z(7)	97,8 %	2,2 %	0 %	0 %	0 %	0 %
Z(6)	99,1 %	0,9 %	0 %	0 %	0 %	0 %
Z(5)	100 %	0 %	0 %	0 %	0 %	0 %
Z(4)	100 %	0 %	0 %	0 %	0 %	0 %
Z(3)	100 %	0 %	0 %	0 %	0 %	0 %
Z(2)	100 %	0 %	0 %	0 %	0 %	0 %
Z(1)	100 %	0 %	0 %	0 %	0 %	0 %
Z(0)	100 %	0 %	0 %	0 %	0 %	0 %
Co(1)	63,9 %	24,8 %	11,3 %	0 %	0 %	0 %
Co(0)	53,5 %	19,3 %	14,0 %	11,2 %	2,0 %	0 %
Ov	29,6 %	15,1 %	13, 3 %	11,0 %	9,8 %	21,2 %

Tabla 4-25: Distribución de los errores para la máxima partición en zonas disjuntas (RTL-disjunto 1)

En la tabla 4-25 se aprecia que para las salidas *Z* el error es mínimo y la práctica totalidad de los errores están por debajo de 0,01. Sin embargo, para las salidas *Ov* y *Co* el error es mayor, especialmente para *Ov*, cuyo máximo error es -0,29. Y aunque este error se da para las ya comentadas condiciones extremas P_{in}=0,5 y a_{in}=1, menos de un 30% de los casos tienen un error por debajo de 0,01.

La distribución del error de la señal *Ov* se muestra en la gráfica 4-25.



Gráfica 4-25: Distribución de los errores absolutos de Ov para el circuito 7 para todo el rango de probabilidades y actividades y con la máxima partición en zonas disjuntas (RTL-disjunto 1)

Ya se ha comentado que para minimizar el error la actividad de la señal de selección debe ser baja. Y es por esto que normalmente salen errores tan altos en la zona cercana a a=1. Si se limita la actividad de la señal de selección se obtienen errores menores (ver el apartado AIII.3.1 del anexo).

Para este circuito la señal de selección es de 6 bits (*Cmd*), por tanto, para que esta señal considerada como vector tenga una actividad baja, es preciso que las actividades individuales de cada índice sean muy bajas.

La actividad de *Cmd* es:

$$a_{Cmd} = a_{Cmd(0)} + a_{Cmd(1)}\cdot\overline{a}_{Cmd(0)} + a_{Cmd(2)}\cdot\overline{a}_{Cmd(1)}\cdot\overline{a}_{Cmd(0)} + a_{Cmd(3)}\cdot\overline{a}_{Cmd(2)}\cdot\overline{a}_{Cmd(1)}\cdot\overline{a}_{Cmd(0)} +$$

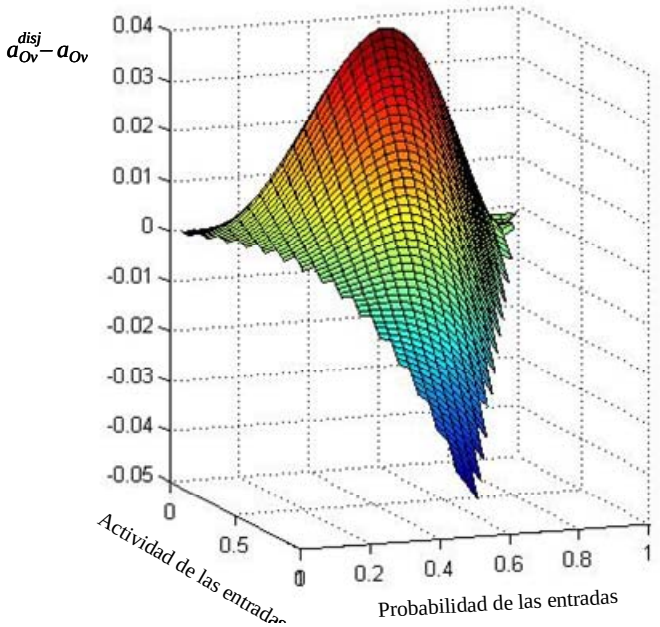
$$+ a_{Cmd(4)} \cdot \overline{a}_{Cmd(3)} \cdot \overline{a}_{Cmd(2)} \cdot \overline{a}_{Cmd(1)} \cdot \overline{a}_{Cmd(0)} + a_{Cmd(5)} \cdot \overline{a}_{Cmd(4)} \cdot \overline{a}_{Cmd(3)} \cdot \overline{a}_{Cmd(2)} \cdot \overline{a}_{Cmd(1)} \cdot \overline{a}_{Cmd(0)}$$

Así, si se limita la actividad de cada índice de *Cmd* que sea menor que 0,1, la actividad de la señal *Cmd* será 0,47 como máximo. La tabla 4-26 muestra cómo queda la distribución de los errores bajo estas condiciones.

distribución	$\|e\| < 0{,}01$	$0{,}01 < \|e\| < 0{,}02$	$0{,}02 < \|e\| < 0{,}03$	$0{,}03 < \|e\| < 0{,}04$	$0{,}04 < \|e\| < 0{,}05$	$\|e\| > 0{,}06$
Z(7)	98,7 %	1,3 %	0 %	0 %	0 %	0 %
Z(6)	99,8 %	0,2 %	0 %	0 %	0 %	0 %
Z(5)	100 %	0 %	0 %	0 %	0 %	0 %
Z(4)	100 %	0 %	0 %	0 %	0 %	0 %
Z(3)	100 %	0 %	0 %	0 %	0 %	0 %
Z(2)	100 %	0 %	0 %	0 %	0 %	0 %
Z(1)	100 %	0 %	0 %	0 %	0 %	0 %
Z(0)	100 %	0 %	0 %	0 %	0 %	0 %
Co(1)	71,4 %	26,4 %	2,2 %	0 %	0 %	0 %
Co(0)	60,6 %	23,9 %	13,2 %	2,3 %	0 %	0 %
Ov	48,6 %	25,2 %	16,7 %	8,6 %	0,9 %	0 %

Tabla 4-26: Distribución de los errores para la partición disjunta exhaustiva (RTL-disjunto 1) limitando la actividad de Cmd ($a_{Cmd(i)} < 0{,}1$).

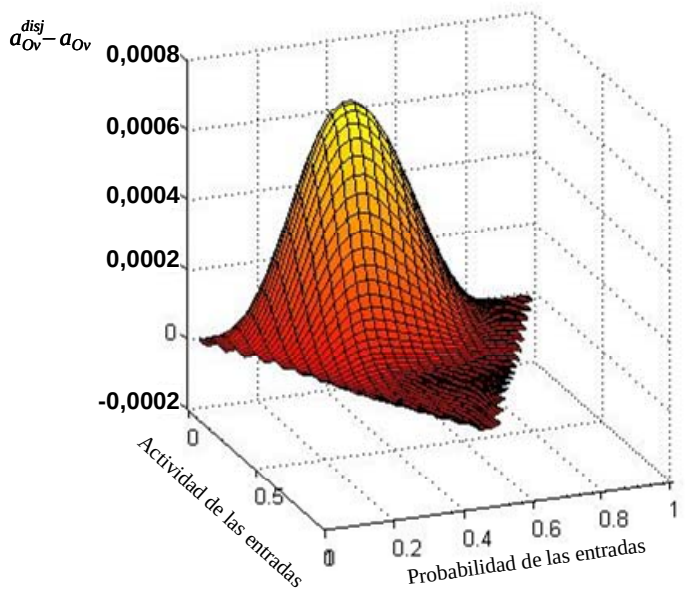
Con esta condición, la distribución de los errores ha mejorado para todas las señales. Aún así, la señal *Ov* tiene errores importantes. El peor caso sigue sucediendo para P_{in}=0,5 y a_{in}=1, y vale -0,046. En la gráfica 4-26 se representan los errores cometidos.



Gráfica 4-26: Distribución de los errores absolutos de Ov para el circuito 7 para todo el rango de probabilidades y actividades y con la máxima partición en zonas disjuntas (RTL-disjunto 1). La limitando la actividad de Cmd está limitada: $a_{Cmd(i)} < 0{,}1$

El análisis del error para la **segunda partición en zonas disjuntas (RTL-disjunto 2)** es muy diferente. Con estas áreas más grandes el error se mantiene más limitado. Ahora, los errores de **todas las señales se mantienen por debajo de 0,01**. En consecuencia **se puede considerar que no se comete error**. Esto es así incluso si no se limita la actividad de conmutación de la señal de selección.

Ahora, la distribución del error para *Ov* se muestra en la gráfica 4-27, donde se ve que es varios órdenes de magnitud menor que el de la partición disjunta exhaustiva (RTL-disjunto 1).



Gráfica 4-27: Distribución de los errores absolutos de Ov para el circuito 7 para todo el rango de probabilidades y actividades y con una partición media en zonas disjuntas (RTL-disjunto 2)

Para la partición RTL-disjunto 2 el error máximo no se produce en *Ov* sino en *Z*(7). Aún así, este error se considera despreciable: e_{max} = -0,0068.

La partición RTL-disjunta 2 no llega a ser la partición disjunta mínima (ver el apartado AIII.3.1 del anexo), y aún así se consiguen errores despreciables. Como conclusión, al realizar la partición en regiones disjuntas se debe evitar dividir el circuito en regiones muy pequeñas, tendiendo a realizar particiones de tamaños grandes que sean manejables por las herramientas de cálculo. **Estas decisiones se pueden tomar durante el proceso de estimación ya que el modelo extendido del hardware** (EHM, §3.3.2) proporciona la información relativa al número de dependencias, y éstas influyen notablemente en el tamaño de los BDD. Procediendo de esta manera **se logran errores bajos**.

4.4 Análisis global

A modo de resumen, en este apartado se dará una visión global y simplificada de los resultados obtenidos respecto a las reducciones de tamaños de los BDD y los errores cometidos. Los análisis que se mostrarán son:

- La reducción del tamaño de los BDD por el uso de los BDD de actividad propuestos (*a*BDD frente a TFBDD)
- La reducción del tamaño de los *a*BDD debido al análisis en RTL frente a puertas
- La reducción del tamaño de los *a*BDD debido a la partición por regiones disjuntas frente al RTL exacto
- La reducción del tamaño de los *a*BDD debido a la partición por regiones disjuntas frente al análisis en puertas.
- El error cometido por la partición por regiones disjuntas cuando suponga una ventaja en el análisis

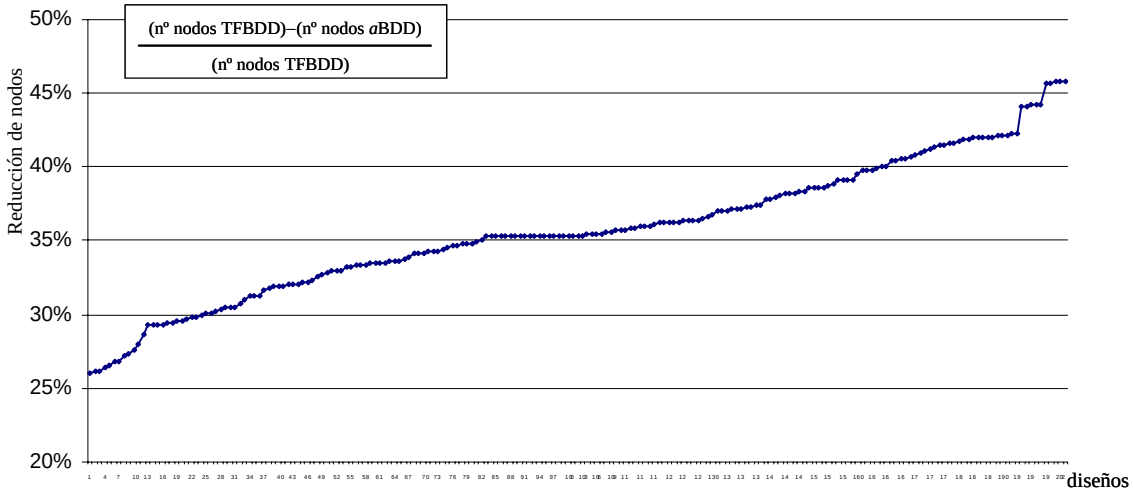
Este análisis pretende dar una visión general y simplificada, por tanto para no sobrecargarlo de información, las comparaciones entre los distintos análisis se harán sólo con los tamaños de los *a*BDD. Las comparaciones con los TFBDD serán del mismo orden de magnitud que los *a*BDD, ya que los TFBDD suelen ser algo más de un 30% mayores. Las diferencias entre los BDD de probabilidad no son tan grandes, sin embargo los BDD de probabilidad no son tan críticos

debido a su menor tamaño, y por tanto no se incluirán en este apartado por su objetivo de simplicidad. Toda la información que no se incluye en este apartado se encuentra en los análisis individuales de cada circuito.

4.4.1 Reducción de nodos por el uso de los *a*BDD

De los 7 circuitos estudiados, se han realizado análisis en puertas, en RTL y en RTL-disjunto. En algunos de ellos se han realizado variaciones en el número de bits de los operandos. Para algunos circuitos se ha realizado más de una síntesis en puertas con distintas opciones y variaciones en el orden de las puertas (como las realizadas para el circuito "max" en el anexo AIII.1.1). En total se han realizado más de 200 análisis, y en todos los casos ha habido una reducción del número de nodos debida al uso de los *a*BDD frente a los TFBDD.

En la siguiente gráfica se muestra el porcentaje de reducción de número de nodos de los BDD de actividad. Para cada diseño, se han sumado todos los nodos de los BDD de actividad necesarios usando *a*BDD por un lado, y TFBDD por otro, y se ha obtenido el porcentaje de reducción de nodos. La gráfica se ha ordenado por la reducción lograda.



Gráfica 4-28: Porcentaje de reducción obtenida por el uso de los aBDD frente a TFBDD

En el extremo inferior de la gráfica 4-28 está el 26% de reducción logrado en el sumador/restador de 15 bits analizado en RTL, que de 12738 nodos usando TFBDD se pasa a 9428 nodos con *a*BDD. En el otro extremo está la reducción lograda en la ALU mediante el análisis RTL-disjunto. En ella se ha pasado de 21421 nodos con TFBDD a 11608 nodos usando *a*BDD (46%).

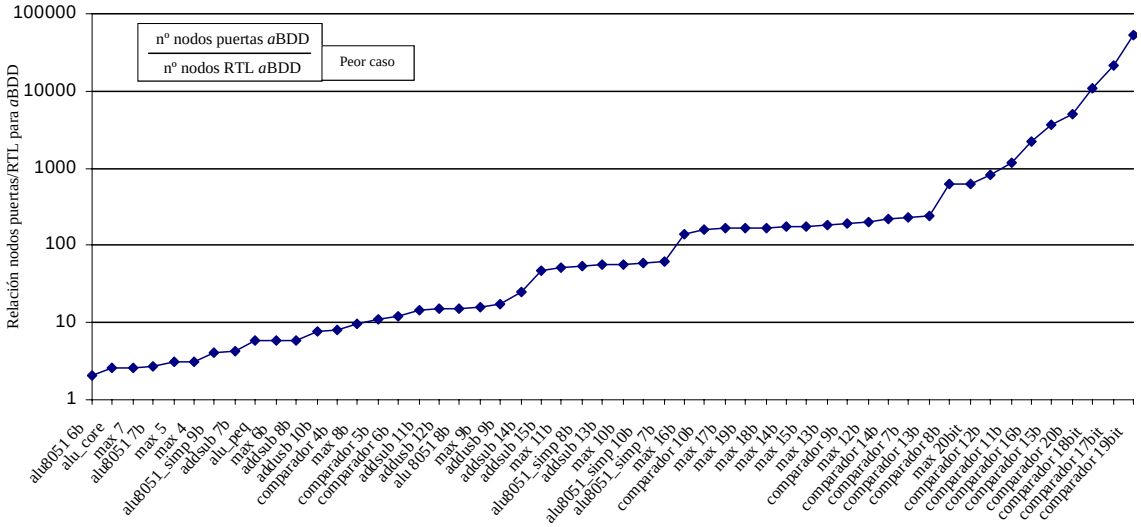
Por tanto, **el uso de los BDD de actividad propuestos disminuye el número de nodos entre un 26% y 46%**, con la ventaja añadida de que **su uso no produce error en el cálculo de actividad**.

4.4.2 Reducción del tamaño de los BDD por el análisis en RTL frente a puertas

El análisis en RTL sin realizar partición por regiones disjuntas frente al análisis al análisis en puertas normalmente produce BDD de menor tamaño. Este análisis en RTL, por no realizar la partición por regiones disjuntas, es exacto. El menor tamaño de los BDD se produce por lograr un mejor ordenamiento y también porque a veces la síntesis en puertas hace aumentar las reconvergencias.

En los BDD de actividad se consiguen grandes diferencias respecto a puertas. En la gráfica 4-29 se muestra la relación entre los nodos de los *a*BDD en puertas respecto a RTL-exacto. Obsérvese

que la escala es logarítmica, y que normalmente las diferencias se incrementan con el aumento del ancho de bus de los operandos (el número de bits).



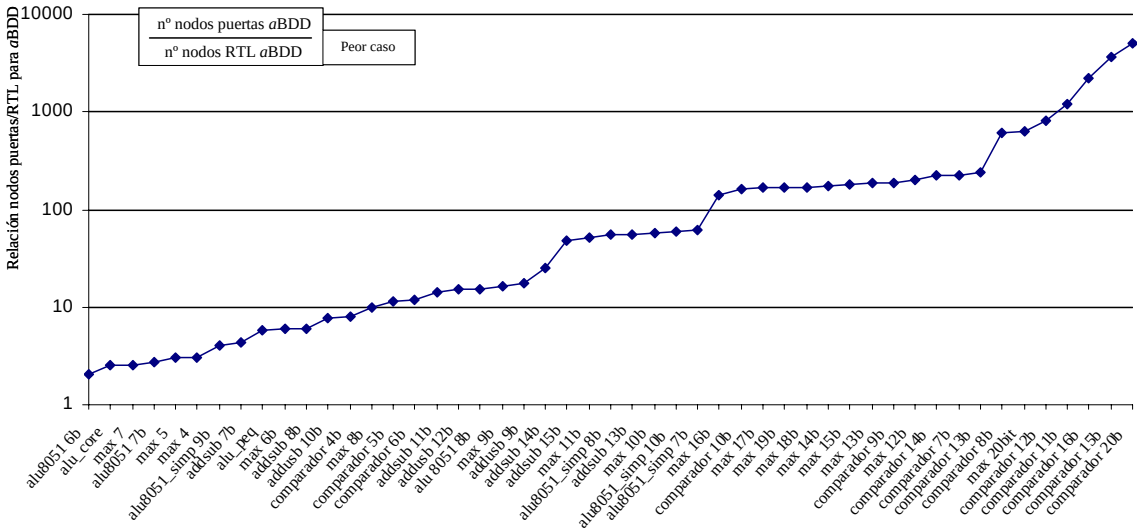
Gráfica 4-29: Relación entre la suma de nodos de los aBDD obtenidos mediante el análisis en puertas frente a la suma de nodos obtenida en RTL (exacto)

La menor relación se tiene con el circuito "comparador" (circuito 2) de cuatro bits (1,74 veces el número de nodos para puertas), mientras que la mayor relación la tiene el mismo circuito pero con 19 bits. En este caso, la suma de nodos de los *a*BDD del análisis en puertas es más de 2000 veces mayor que en RTL. Esto también demuestra la influencia del ancho de bus en las diferencias de tamaño de los *a*BDD de RTL respecto a los de puertas.

Sin embargo, lo más interesante son los análisis que no aparecen en la gráfica, pues son aquellos que en puertas no se pudieron realizar por superar los límites de memoria del ordenador, y que sin embargo, sí se pudieron realizar en RTL. Estos casos fueron:

- El circuito "max" (circuito 1) a partir de 21 bits, o incluso con menos bits según el caso (véase el apartado AIII.1 anexo)
- El circuito "comparador" (circuito 2) a partir de 20 bits
- La ALU simplificada del 8051 (circuito 6) a partir de 11 bits
- La ALU completa del 8051 (circuito 7) a partir de 9 bits

La comparación entre el *a*BDD de mayor tamaño en análisis en puertas y RTL también es favorable para RTL. El tamaño de este *a*BDD es muy importante, ya que puede determinar si se puede llevar a cabo el análisis o no. La gráfica 4-30 muestra las relaciones entre estos tamaños según los circuitos. Donde se observa que las relaciones son mayores que para la suma total de nodos y que también normalmente se incrementan cuando se aumenta el número de bits de los operandos.

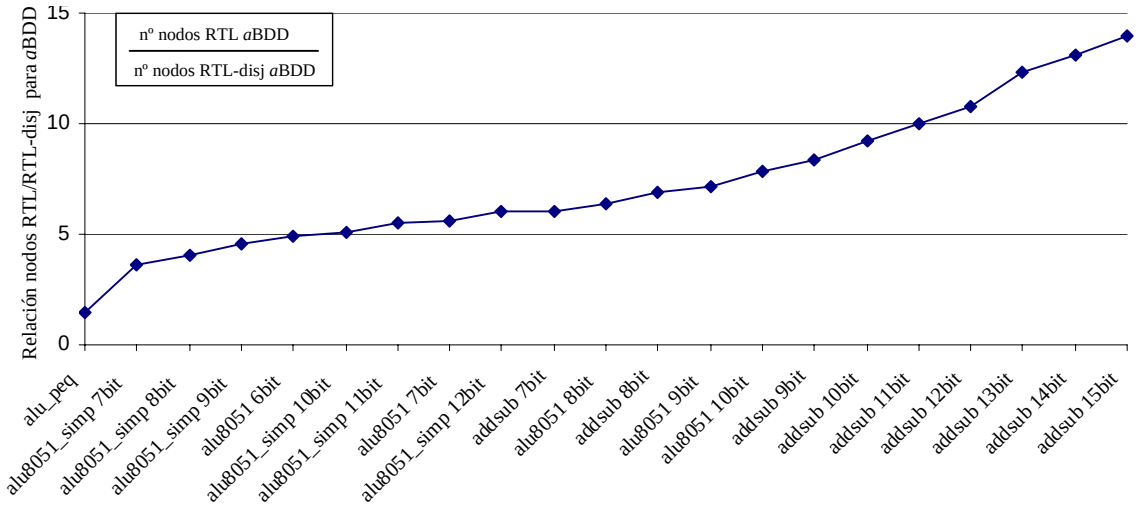


Gráfica 4-30: Relación entre el tamaño del mayor aBDD obtenido mediante el análisis en puertas frente al obtenido en RTL (exacto)

4.4.3 Reducción del tamaño de los BDD por el análisis en RTL-disjunto frente a RTL

Ya se ha comentado que la partición RTL-disjunta se debe realizar sólo cuando el tamaño de los BDD sea grande y en las condiciones en que se espere que el error provocado no sea elevado. Gracias al modelo extendido del hardware (EHM §3.3.2) estas condiciones se pueden evaluar durante el análisis RTL.

En la gráfica 4-31 se muestra la relación entre la suma de nodos de los *a*BDD obtenidos del análisis en RTL exacto frente a la suma de nodos obtenida del RTL disjunto. En la gráfica no se incluye el circuito "max", ya que para él no compensa realizar la partición disjunta. Tampoco se incluye el circuito "comparador", puesto que la partición disjunta era equivalente a la partición reconvergente. Por último, tampoco se han incluido las particiones disjuntas exhaustivas. Normalmente estas particiones son más ventajosas en número de nodos, pero como se ha visto, producen un error más elevado (véase apartado AIII.3.1 del anexo).

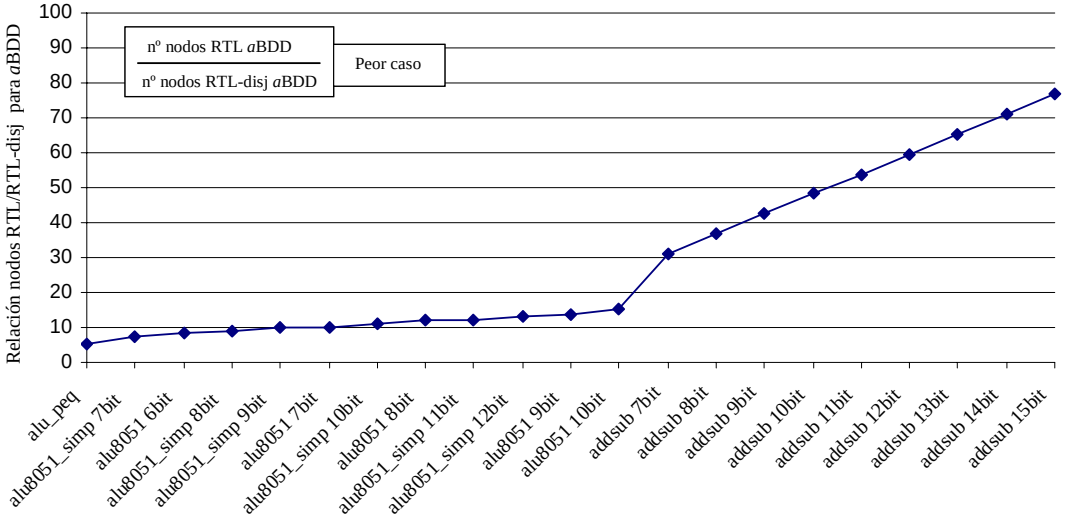


Gráfica 4-31: Relación entre la suma de nodos de los aBDD obtenidos mediante el análisis en RTL (exacto) frente a la suma obtenida en RTL disjunto

En la gráfica 4-31 se puede apreciar cómo las diferencias aumentan cuanto más grande es el circuito. La menor diferencia se da en el circuito "alu_peq" (§4.2.3), en este circuito la suma de

nodos en RTL es aproximadamente 1,5 veces mayor que en RTL-disjunto. La razón de esta baja relación se debe al menor tamaño del circuito, ya que los operandos circuito sólo tienen cuatro bits.

La comparación entre los **BDD de mayor tamaño** es ventajosa para la partición en regiones disjuntas porque las áreas no aumentan con el número de bits, cosa que sí ocurre si no se hace partición. La gráfica 4-32 muestra la relación entre el número de nodos del *a*BDD de mayor tamaño obtenido del análisis RTL exacto y el *a*BDD de mayor tamaño obtenido de la partición disjunta mínima.



Gráfica 4-32: Relación entre el tamaño del mayor aBDD obtenido mediante el análisis en RTL (exacto) frente al obtenido en RTL disjunto

En los casos que se ha analizado la variación del tamaño del mayor *a*BDD con respecto al número de bits de los operandos, se ha visto que **en RTL disjunto el tamaño permanece constante** y que **en RTL exacto el tamaño aumenta linealmente con el número de bits** (en puertas aumenta de manera variable e imprevisible).

A continuación, en la tabla 4-27, se muestra el tamaño de estos *a*BDD para 8 bits y la variación con el número de bits según si se realiza la partición disjunta o no.

Circuito	RTL exacto		RTL disjunto mínimo	
	N de nodos del mayor *a*BDD para 8 bits	N de nodos del mayor *a*BDD para n bits	N de nodos del mayor *a*BDD para 8 bits	N de nodos del mayor *a*BDD para n bits
max[53]	110	$15 \cdot n - 10$	19	19
comparador	12	12	12	12
addsub	479	$74 \cdot n - 113$	13	13
alu8051_simp	3215	$423 \cdot n - 169$	370	370
alu8051	10807	$1467 \cdot n - 929$	912	912

Tabla 4-27: Tamaños de los mayores aBDD para 8 bits de ancho de bus de los operandos y sus variaciones de tamaño con el número de bits de los operandos según se realice o no la partición por regiones disjuntas

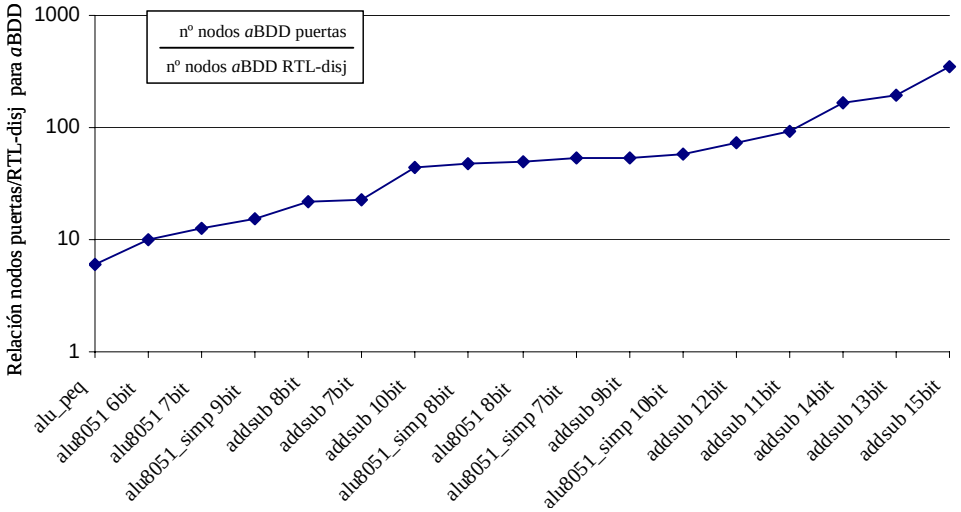
Como se ve en la tabla 4-27, cuanto más complejo es el circuito, mayor es el aumento del *a*BDD de mayor tamaño en RTL exacto. Si bien es muy conveniente lograr un aumento lineal, al contrario de lo que sucede en puertas, podría suceder que para circuitos muy complejos se llegue a tamaños excesivos. Esto puede hacer que **sea necesario realizar la partición por regiones disjuntas** mínima.

[53] El circuito "max" se incluye aunque en él no se realiza la partición disjunta mínima

4.4.4 Reducción del tamaño de los BDD por el análisis en RTL-disjunto frente a puertas

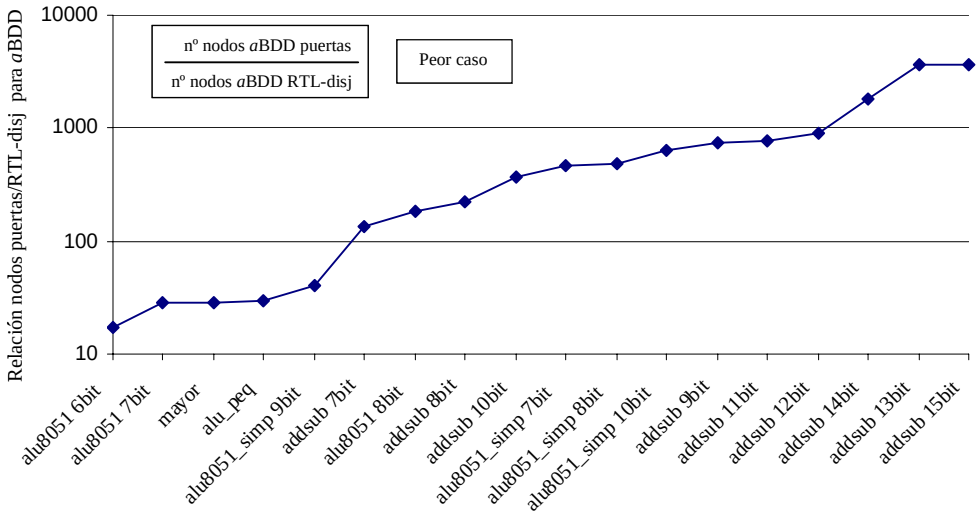
En los anteriores apartados se ha visto que por lo general el análisis en RTL produce tamaños de BDD menores que en puertas, y que el análisis en RTL disjunto produce tamaños de BDD menores que en RTL. Por lo tanto, es de prever mayores diferencias entre los tamaños de los BDD procedentes del análisis en RTL disjunto y los BDD del análisis en puertas. Las gráficas de este apartado demuestran que esto es así.

La gráfica 4-33 muestra la relación entre el número de nodos de los *a*BDD obtenidos en el análisis en puertas frente al obtenido con el análisis en RTL disjunto. Nótese que la gráfica se ha puesto en escala logarítmica para facilitar la visualización.



Gráfica 4-33: Relación entre la suma de nodos de los aBDD obtenidos mediante el análisis en puertas frente a la suma obtenida en RTL disjunto

A continuación se muestra la relación entre los tamaños de los mayores *a*BDD de cada circuito en que se ha realizado la partición por regiones disjuntas (gráfica 4-34).



Gráfica 4-34: Relación entre el tamaño del mayor aBDD obtenido mediante el análisis en puertas frente al obtenido en RTL disjunto

Al la vista de las gráficas, la ventaja en cuanto a tamaños de los BDD por el uso del análisis en RTL-disjunto frente a puertas es evidente.

4.4.5 Error cometido por el método propuesto

El método propuesto tiene dos variantes dependiendo de si se aplica la partición por regiones disjuntas o no. Cuando no se realiza la partición por regiones disjuntas el método es exacto. En todos los experimentos realizados en el nivel RT se ha podido realizar el cálculo de la actividad sin necesidad de realizar la partición disjunta. Al contrario de lo que ocurre en el nivel de puertas, en donde en algunos casos ha sido imposible realizar los cálculos debido al gran tamaño de los BDD. Por tanto, **con los circuitos analizados se puede calcular la actividad de manera exacta mediante el método propuesto**. Sin embargo, la partición por regiones disjuntas aporta una significativa reducción en los tamaños de los BDD y es de esperar que para circuitos de mucho mayor tamaño incluso en el nivel RT sea necesario recurrir a la partición disjunta para evitar un excesivo tamaño de los BDD.

En este apartado se mostrará el error producido en cada circuito analizado. No se mostrará el error producido por la partición disjunta exhaustiva, ya que esta partición no es recomendable. En secciones anteriores se han mostrado los resultados de la partición disjunta exhaustiva para demostrar que no es adecuada, pero no como propuesta de partición.

La tabla 4-28 muestran las estadísticas de los errores debidas a las particiones disjuntas en cada uno de los circuitos. Los errores máximos están en valor absoluto ($|e_{max}|$). En los circuitos más pequeños ("max", "comparador" y "alu_core") no se han realizado las particiones por el pequeño tamaño de los BDD resultantes, por lo tanto, su error es nulo.

Circuito		Peor caso *a*BDD		Estadísticas de los errores			Comentarios
nº	nombre	RTL-exacto	RTL-disjunto	μ	σ	$\lvert e_{max} \rvert$	
1	max	50	-	0	0	0	sin partición disjunta
2	comparador	12	-	0	0	0	sin partición disjunta
3	alu_peq	438	85	0,0080	0,0143	0,109	situaciones irreales
4	alu_core	168	-	0	0	0	sin partición disjunta
5	addsub	479	13	0,0011	0,0049	0,125	situaciones irreales
6	alu8051_simp	3215	370	0,0011	0,0019	0,012	situaciones irreales
7	alu8051	10807	912	0,00056	0,00088	0,0068	situaciones irreales

Tabla 4-28: Reducciones en el tamaño de los aBDD y estadísticas de los errores debidas a la partición disjunta

En la tabla se puede observar que, en general, cuanto menor es el tamaño de los *a*BDD del RTL exacto, mayor son los errores. Los mayores errores los tienen "alu_peq" y "addsub", cuyos tamaños de *a*BDD del análisis RTL-exacto son 438 y 479 nodos respectivamente. Mientras que "alu8051_simp", con 3215 nodos en el *a*BDD del análisis RTL-exacto y 370 nodos en el *a*BDD del análisis RTL-disjunto, tiene errores más bajos. El error medio de "alu8051_simp" es igual que el de "addsub", sin embargo, su desviación típica y error máximo son mucho menores. Por último, el circuito "alu8051" tiene errores despreciables, incluso su error máximo.

Estos resultados parecen sugerir que los circuitos "alu_peq" y "addsub" son todavía pequeños para realizar la partición disjuntas. Incluso para "alu8051_simp" podría no realizarse la partición. Y sólo para circuitos del tamaño de "alu8051" o superiores se debería realizar la partición. Esta decisión se puede tomar durante el análisis del circuito, ya que el EHM ofrece información que permite predecir el tamaño de los BDD resultantes.

Sin embargo, aunque los circuitos " alu_peq" y "addsub" tengan algunos errores importantes, ya se ha dicho que muchos de esos errores se deben a situaciones extremas de probabilidad y actividad en las entradas. Estas situaciones son irreales en el funcionamiento normal del circuito y son las que hacen empeorar los resultados del error. Durante el análisis del error se realizaron variaciones de los experimentos para evitar esas situaciones improbables. La tabla AIII-18 del anexo amplía la información de la tabla 4-28 incluyendo dichos experimentos. Como se puede apreciar, en tales condiciones los errores se reducen considerablemente.

A raíz de esto último que se acaba de comentar, se debe señalar que los errores del circuito "alu8051" ni siquiera ven afectados por dichas condiciones improbables. Siendo su error máximo inferior a 0,01.

Por último, la tabla 4-28, o si acaso la tabla AIII-18 del anexo con los experimentos corregidos, se podría comparar con la tabla 4-9, donde se muestran los errores de otras propuestas. Ya se dijo en la sección 4.3.1 que los resultados no son directamente comparables, pero por lo menos podrá servir para tener una referencia.

4.5 Tiempos de procesamiento

Los análisis se han realizado mediante una herramienta informática cuyas características se explican con más detalle en el anexo IV. El programa que lleva a cabo el análisis probabilístico se encuentra en una fase inicial, y por tanto no está optimizado. Este programa es complejo y de gran tamaño, ya que, entre otras funciones, crea el EHM, analiza las reconvergencias, realiza la partición del circuito, construye los BDD, calcula las actividades y las propaga. Para la creación del programa se han necesitado de muchas revisiones para su depuración, ya que en el análisis de circuitos grandes se había visto que el programa incurría en redundancias que hacían que fuese excesivamente lento. El código ya es bastante más eficiente, aunque todavía necesita seguir siendo depurado y optimizado en gran medida. Por tanto, los resultados de tiempos no son fiables.

Para el análisis se ha utilizado un ordenador con procesador Intel Pentium 4 a 3,20 GHz y 1 GB de memoria RAM. El programa final se ejecuta sobre Cygwin [32], que al ser un emulador de Linux sobre Windows®, los programas tardan mucho más que si fuesen ejecutados directamente en Linux sobre un ordenador de las mismas características. También hay que decir que la medida de tiempos de Cygwin no es exacta, ya que entre varias ejecuciones del mismo programa puede haber diferencias de hasta un 50%.

En la tabla 4-29 se muestran los tiempos de procesamiento para el circuito "alu8051" (circuito 7). Se ha mostrado este circuito por ser el más complejo. El resto de circuitos muestran tiempos menores. Los circuitos "alu8051" y "alu8051_simp" son los únicos que en RTL y RTL-disjunto tardan más de un segundo en el procesamiento, en los demás circuitos y para cualquier número de bits de los analizados, el procesamiento dura menos de un segundo.

tiempo (s)	Puertas	RTL	RTL-disjunto
6 bits	830 (13' 50")	4	3,5
7 bits	7494 (2h 4' 54")	5	4
8 bits	21536 (5h 58' 56")	6	4,5
9 bits	*sin memoria*	7	5
10 bits	*sin memoria*	8	5,5

Tabla 4-29: Tiempos de procesamiento en segundos para el circuito "alu8051" (circuito 7) según el ancho del bus y el tipo de procesamiento

Los datos de la tabla 4-29 deben tomarse con cautela, ya que, como se ha dicho, el código ha tenido que ser depurado en muchas ocasiones. Antes de estas correcciones, los tiempos de procesamiento han llegado a ser varios órdenes de magnitud superiores a los que se muestran en la tabla: el análisis en puertas llegaba a durar más de una semana y en RTL varias horas. Actualmente se han detectado partes del código que todavía pueden ser optimizadas, y que probablemente harían reducir enormemente el tiempo procesamiento. Estas optimizaciones no se han llevado a cabo debido a que, en las condiciones actuales, el tiempo de procesamiento en RTL se considera aceptable. En consecuencia, **el análisis propuesto es viable en la práctica**.

Se podría haber optimizado el programa de manera que se hubiesen demostrado de manera fiable las diferencias de procesamiento entre los niveles de puertas y RT. Sin embargo, estas modificaciones implican un tiempo de desarrollo elevado debido al gran tamaño del código

(varias decenas de miles de líneas). Este desarrollo no se considera imprescindible para demostrar los beneficios del nivel RT frente a puertas, ya que los resultados experimentales que comparan los tamaños de los BDD se consideran suficientes, junto con la constatación de que en puertas, a diferencia con RTL, hay circuitos que son difícilmente analizables o no se pueden analizar.

También se ha querido obtener una **comparación entre los métodos probabilísticos** (como el utilizado en esta tesis §2.1.2) **con los método simulativos** (§2.1.1). Para ello se han analizado los dos primeros circuitos (circuitos "max" y "comparador" originales, con 4 bits de ancho). Para el análisis se han realizado simulaciones sin retardos del circuito con Modelsim [85] que incluía un programa que monitoriza los estados de las señales y sus transiciones. Para cada circuito la simulación duró más de 4 minutos para 16384 (2^{14}) vectores de test en el nivel RT, mientras que la misma simulación con los circuitos en puertas sin retardos duraron casi 10 minutos. El análisis probabilístico de estos circuitos duró unos 0,2 segundos. Para el análisis probabilístico, en estos circuitos el tiempo de procesamiento es similar en puertas o en RTL, ya que por el pequeño tamaño del circuito, no hay mucha diferencia entre los tamaños de los BDD. Por tanto, en estos ejemplos se constata la ventaja del método probabilístico frente al método simulativo. Estas ventajas aumentan con el tamaño de los circuitos, ya que las simulaciones son más largas.

Por último añadir que una de las principales ventajas del método probabilístico es que es un análisis de una vez. Esto es, una vez que se ha creado el modelo probabilístico, si se cambian las condiciones de entrada, no hace falta rehacer el modelo[54], y por lo tanto, los tiempos de análisis pueden ser un orden de magnitud menor.

4.6 Conclusiones a los resultados experimentales

Este capítulo de resultados no sólo ha permitido demostrar de manera general la ventaja del análisis de la actividad en el nivel RT frente al análisis en puertas, sino que también ha servido para confirmar las ventajas de muchas de las propuestas individuales realizadas en esta tesis doctoral.

De manera general **se ha demostrado la conveniencia del análisis en el nivel RTL frente al nivel de puertas**. El análisis en RTL sin partición por regiones disjuntas **no produce ningún error de cálculo** y sin embargo, se ha visto que tiene las siguientes ventajas frente al nivel de puertas:

- **El análisis en RTL proporciona un ordenamiento adecuado**. A diferencia que en puertas, donde se obtienen ordenamientos variables y poco predecibles. Las gráficas del apartado 4.4.2 comparan los tamaños y demuestran la validez de la propuesta del ordenamiento RTL (§3.6). En la mayoría de los casos analizados, la suma de los nodos de los BDD de actividad es más del doble en puertas que en RTL, en algunos casos incluso llegan a ser más de mil veces mayor en puertas que en RTL. Este ordenamiento se consigue tanto por el análisis de las estructuras RTL del circuito y el conocimiento de las operaciones involucradas, como por la localización de los índices de los operandos.
- **Las diferencias aumentan en favor del análisis RTL con el tamaño y el número de entradas de los circuitos**. Al aumentar el ancho de bus de los circuitos, llega un momento en que por el elevado tamaño de los BDD en el **nivel de puertas**, **no se pueden analizar** los circuitos, aunque sí se pueden analizar desde el nivel RT.
- En los circuitos analizados, al aumentar el ancho de bus de los operandos, los BDD de mayor tamaño aumentan de forma lineal en el análisis RTL-exacto, mientras que en puertas aumentan de manera imprevisible y con una tendencia superior a la lineal.

[54] Esto es cierto a no ser que se hayan realizado particiones disjuntas, y que con las nuevas condiciones de probabilidad y actividad no sea aconsejable realizar la partición. Aún así, gran parte del análisis estaría hecho.

- **El proceso de síntesis a veces incrementa la reconvergencia de los circuitos**. Esto hace que se resulten regiones de mayor tamaño para el análisis en puertas. Recuérdese que la partición por regiones reconvergentes no produce error, por tanto, es una clara ventaja del análisis RTL ya que se consiguen BDD de menor tamaño sin coste en la exactitud del análisis.
- **El análisis en RT permite realizar una partición en zonas disjuntas de manera sencilla**, mientras que en el nivel de puertas la detección de estas zonas es complicada. Los resultados obtenidos con la partición disjunta no son exactos, pero se deja la opción de realizar la partición o no según se cumplan las condiciones propuestas para minimizar el error y sea el tamaño de las regiones.
- El nivel RT facilita la comprensión de la funcionalidad del circuito, permitiendo un análisis que posibilite una partición por funcionalidad (ver anexos AIII.5.1 y AIII.7.1).

Además de estas ventajas generales, se han demostrado las ventajas de las propuestas particulares de esta tesis:

- La propuesta de representación de los BDD de actividad (*a*BDD §3.5.2) **en todos los casos produce BDD de menor tamaño**. En el apartado 4.4.1 se mostró que las reducciones iban desde un 26% hasta un 46%. Estos experimentos corroboran los estudios teóricos expuestos en el apartado 3.5.
- Para los circuitos grandes, la **partición por regiones disjuntas produce BDD de menor tamaño**, tanto comparados con los BDD provenientes del análisis en RTL exacto (§4.4.3) como los BDD del análisis en puertas (§4.4.4). La diferencia es mayor para los BDD de actividad, que son precisamente los que, por su mayor tamaño, son más críticos.
- **Las diferencias aumentan en favor del análisis RTL disjunto con el tamaño y el número de entradas de los circuitos**. En los experimentos realizados, en RTL disjunto las dimensiones de los BDD de mayor tamaño permanecen constantes al aumentar el ancho de bus de los operandos, mientras que en RTL exacto el aumento suele ser lineal. Por otro lado, en puertas, el aumento es impredecible y con una tendencia superior a la lineal, haciendo que llegado a un determinado ancho de bus, el circuito no se pueda analizar en puertas.
- El **modelo extendido del hardware** (§3.3.2) proporciona un medio eficaz para identificar las regiones disjuntas y analizar si se cumplen las condiciones que minimizan el error (§3.4.3). Para reducir el error es conveniente evitar las particiones disjuntas exhaustivas y **realizar particiones disjuntas mínimas**.
- En consecuencia, **el error producido por la partición por regiones disjuntas puede preverse** según las características del circuito, la partición realizada y las condiciones de probabilidad y actividad de las señales involucradas. Hay por tanto, una **característica estructural** y otra **circunstancial**. En el apartado del análisis del error (§4.3) se vio que éste se puede minimizar, y que el error depende de las condiciones que se expusieron en el apartado 3.4.3 (y anexo AII.2).
- Manteniendo las condiciones recomendadas para la partición y si no hay condiciones de probabilidad y actividad extremas, **el error producido por el uso de regiones disjuntas se mantiene en valores aceptables** (por debajo de 0,01).
- Se ha comentado (§4.1) que por las limitaciones del programa informático de análisis, no se ha podido realizar análisis para circuitos muy grandes y complejos. Sin embargo, la tendencia observada en los análisis permite pronosticar que la partición disjunta mínima será muy útil en circuitos grandes y complejos. En estos circuitos, una partición disjunta mínima permitirá reducir el tamaño de las regiones y mantener un error pequeño (como sucedió con el circuito "alu8051", el circuito 7). Estas regiones más pequeñas no serían posibles de conseguir en RTL-exacto, y si la complejidad del circuito es elevada, pudiera suceder que en RTL-exacto los tamaños de los BDD resulten muy elevados como para poder realizar el análisis.

Para concluir, este capítulo ha permitido demostrar que **el análisis de la actividad de conmutación en RTL es más favorable que en puertas**. El análisis en RTL permite trabajar con BDD más pequeños sin perder exactitud respecto a puertas. Las diferencias de tamaños se deben principalmente a un mejor ordenamiento de los BDD debido al seguimiento de la ordenación RTL propuesta en esta tesis. Los tamaños de los BDD de RTL pueden llegar a ser hasta 1000 veces menores que en puertas, y lo que es más importante, para los circuitos grandes o complejos, los tamaños de los BDD procedentes del nivel de puertas pueden ser tan grandes que no resulte factible el análisis, mientras que en RTL sí.

Esto ha permitido que el método propuesto sea el **único** de todos los analizados (tabla 2-2) **que pueda realizar el análisis exacto de la actividad** de circuitos de tamaño mediano. Quedaría por analizar el comportamiento del método propuesto para circuitos de tamaño mayor.

Además, dentro del análisis RTL, para circuitos grandes y complejos, el tamaño de sus BDD se puede reducir aún más realizando particiones disjuntas mínimas. Estas particiones, llevadas a cabo siguiendo las condiciones de partición propuestas en esta tesis, producen un error que permanece dentro de unos límites aceptables.

5. CONCLUSIONES

Esta tesis doctoral aborda la estimación probabilística de la actividad de conmutación durante el diseño de los circuitos electrónicos digitales. La actividad de conmutación es un parámetro fundamental para la evaluación del consumo dinámico de los circuitos digitales y su estimación temprana puede ayudar a reducir las iteraciones en el flujo del diseño. Estas iteraciones en el flujo del diseño aumentan los costes y producen retrasos en la puesta en mercado del producto.

Además, el análisis probabilístico de la actividad de conmutación de un circuito puede ser útil para otros estudios como son los análisis de testabilidad, fiabilidad, calidad, verificación y análisis comportamentales del circuito.

La propuesta principal de esta tesis doctoral es subir el nivel de abstracción de la estimación de la actividad de conmutación desde el nivel de puertas al nivel RT. Con esto se logran dos objetivos, el primero metodológico y segundo práctico, estos objetivos son:

- Anticipar la información ofrecida al diseñador.
- Aprovechar las características del nivel RT para optimizar los métodos de estimación.

Del primer objetivo se consigue que el **diseñador obtenga con anterioridad una información** que le **permitirá tomar decisiones de diseño**. Cuanto antes en el flujo del diseño se tomen estas decisiones **menos iteraciones en el flujo de diseño** se realizarán. Por tanto, este objetivo es metodológico.

El segundo objetivo es práctico. Con éste se pretende **aprovechar las ventajas** de realizar la estimación en un nivel de abstracción superior para reducir la complejidad del análisis. Las principales características del nivel RT que pueden aprovecharse para optimizar el método de estimación son:

- **Mayor simplicidad** de la descripción, ya que cuanto más alto sea el nivel de abstracción, menos información de detalle se tiene. Esta información, aunque en ocasiones pueda ser útil, provoca una mayor complejidad del análisis.
- **Mayor información** de alto nivel. El nivel RT proporciona información relativa al funcionamiento del circuito que es muy difícil de extraer desde el nivel de puertas. Hacer uso de esta información **facilita enormemente** la estimación. Ejemplos de este tipo de información son:
    - La existencia de operadores lógicos y aritméticos.
    - La definición de estructuras como los multiplexores.
    - Los agrupamientos de señales en vectores y la existencia de tipos de datos enteros y enumerados.
- **Menor reconvergencia**. En algunos casos, las reconvergencias de un circuito pueden aumentar al realizar la síntesis lógica. El aumento de las reconvergencias hace que el número de regiones de reconvergencia disminuya, provocando un mayor tamaño de estas regiones y de los BDD asociados a éstas. Como la partición en regiones reconvergentes no produce error en el cálculo, esto es una clara ventaja del análisis RTL frente al análisis en puertas.

Por tanto, en esta tesis **se ha propuesto un método probabilístico de estimación de la actividad en el nivel RT** que aprovecha las ventajas que este nivel ofrece. Conceptualmente, el método propuesto es una ampliación de los métodos existentes en el nivel de puertas. En consecuencia, los circuitos descritos en el nivel de puertas también pueden ser analizados con este método, aunque sin las ventajas que el nivel RT aporta. Por otro lado, la subida del nivel de abstracción también implica la aparición de dificultades. Todo esto ha motivado a realizar propuestas que aprovechan las ventajas y superan los obstáculos del nivel RT, a la vez que se han realizado propuestas que son válidas tanto en puertas como en RTL. Estas propuestas son:

- Se ha propuesto **la elaboración de un modelo extendido del hardware** (EHM) que proporciona una estructura sobre la que se sustenta la estimación probabilística. Este modelo

permite asemejar el nivel RT a la estructura de *netlist* del nivel de puertas, consiguiendo una estructura de dependencias e influencias entre las señales del circuito. Sin embargo, este modelo conserva la simplicidad del nivel RT y la información relativa a este nivel. Este modelo permite realizar la partición del circuito y construir los BDD por medio de los que se calculan las actividades y probabilidades de las señales. A través de este modelo se propagan las actividades y probabilidades desde las entradas a las salidas.

- Se ha propuesto **una manera eficiente de ordenar las variables de los BDD** asociada al nivel RT. El tamaño de los BDD es fuertemente dependiente del orden de sus variables y, en el nivel de puertas, no es fácil obtener un orden adecuado. Sin embargo, se ha observado que siguiendo un orden *natural* proporcionado por la estructura RTL y con el conocimiento de los índices de los vectores y los operadores involucrados, se consigue un ordenamiento bastante optimizado de los BDD.
- Se ha propuesto realizar una **partición por regiones disjuntas** en los circuitos en los que su tamaño impida un análisis eficiente. En esta tesis se ha propuesto una manera sencilla de realizar estas particiones en el nivel RT. Por el contrario, estas particiones no son fáciles de llevar a cabo en el nivel de puertas. Aunque esta partición conlleva un cierto error en el cálculo de la actividad, en esta tesis se ha propuesto una manera de realizar la partición en la que se minimiza el error.
- Se ha propuesto una **nueva representación de los BDD de actividad** (*a*BDD) que, sin producir error, reduce el número de nodos entre un 25% y un 50%. Esta representación es válida tanto para el análisis en el nivel RT como para el nivel de puertas. Para la construcción de estos BDD, en esta tesis se han definido las operaciones implicadas en su creación y manipulación, además de la transformación de los anteriores BDD de actividad (TFBDD) en los *a*BDD.

Todas estas propuestas han sido implementadas en una herramienta informática que demuestra su viabilidad y la validez de la propuesta principal de esta tesis: la estimación de la actividad en el nivel RT.

Los resultados experimentales han mostrado que, **en muchos casos**, **la estimación en el nivel de puertas no es factible**, mientras que en el **nivel RT, a través de las propuestas de esta tesis, resulta perfectamente viable**. Para los diseños de tamaño mediano, en los que el análisis se puede llevar a cabo tanto en puertas como en RTL, se han observado claras ventajas respecto al tamaño de los BDD generados. Estas ventajas, que cuantitativamente pueden llegar a ser de **varios órdenes de magnitud**, se deben tanto al mejor ordenamiento de los BDD como a la posibilidad de realizar particiones, tanto reconvergentes como disjuntas.

Aunque no se ha podido comprobar con los mismos circuitos de otras propuestas, análisis con circuitos de tamaño y características similares han mostrado que **el método propuesto es el único que ha podido realizar el estudio de la actividad de conmutación de manera exacta** para circuitos de tamaño mediano.

En el caso que se realicen particiones disjuntas, si se siguen las indicaciones para la partición propuestas en esta tesis, en la mayoría de los casos el error en el cálculo de la actividad se encuentra por debajo de 0,01 en términos absolutos. Dentro de la propuesta de esta tesis se incluyen estudios que se pueden realizar durante la propia partición y permiten valorar el impacto de la partición en el error. En consecuencia, se podrá evaluar si conviene realizar la partición y en qué zonas es más conveniente. Las reducciones en los tamaños de los BDD de actividad logradas por llevar a cabo este tipo de partición propuesta son aún mayores que las logradas por el análisis en RTL, llegando incluso a superar un orden de magnitud. En consecuencia, las reducciones respecto al análisis en puertas se añaden a las ya logradas con el análisis en RTL sin partición.

Como conclusión, el método propuesto para la estimación de la actividad de conmutación proporciona un **medio eficaz para realizar la estimación con anterioridad** en el flujo del diseño. Este método, por aprovechar las características relativas al nivel RT resulta **más sencillo**, requiere **menos recursos computacionales.** Y en caso de circuitos de gran tamaño,

puede realizar particiones disjuntas que, con **un coste pequeño en la exactitud, permite abordar el análisis** de estos circuitos con una menor carga de recursos computacionales.

En cuanto a las **líneas futuras** de esta tesis doctoral, éstas se pueden agrupar en cuatro conjuntos de posibilidades:

1. Relativas al **mejor aprovechamiento que el nivel RT ofrece**.
2. Relativas **a la mejora en la implementación** del método propuesto.
3. Relativas a la **ampliación del método propuesto** en aspectos que no se han considerado.
4. Relativas al aprovechamiento del método propuesto en la **aplicación a otros análisis** del diseño de circuitos digitales.

A continuación se amplía la información sobre estos cuatro conjuntos de posibles líneas futuras:

1. En lo relativo a un **mejor aprovechamiento de las posibilidades del nivel RT**, en primer lugar se podría realizar un estudio del ordenamiento BDD óptimo de los operandos de cada operador RTL. Como la información de los operandos y operadores involucrados se encuentra disponible en el EHM, se construirían los BDD con un orden óptimo. Esto ya se ha realizado en esta tesis para algunos operadores, como son: *mayor que*, *suma*, *resta*. Habría pues que ampliarlo a otros operadores, como por ejemplo la multiplicación y la división. En el capítulo de resultados se ha visto que gracias a este ordenamiento se consiguen grandes diferencias de tamaños respecto a los BDD obtenidos en puertas.

   Otro estudio interesante sería aprovechar la información que el nivel RT ofrece para realizar un análisis del circuito con el fin de extraer su funcionalidad, lo que a su vez podría simplificar posteriores análisis. Ejemplos de estos análisis se muestran en los apartados AIII.5.1 y AIII.7.1 del anexo. En los circuitos de estos apartados se muestran distintos niveles de abstracción dentro del RTL. El análisis funcional de un circuito podría ayudar a obtener un nivel de abstracción más elevado del circuito, lo que ayudaría a la simplificación del análisis. En los ejemplos citados, ese análisis se ha realizado de manera manual, pero el objetivo sería poder realizarlo de manera automática.

   Otra parcela donde el nivel RT puede aportar importantes ventajas es en la consideración de las probabilidades y actividades a nivel vectores, números enteros y tipos enumerados. Frecuentemente, las señales de más de un bit tienen probabilidades y actividades referidas a la señal considerada como un conjunto y tomarlas bit a bit implica una pérdida de información. Por poner como ejemplo el caso extremo de un contador ascendente de 0 a 3, la secuencia de transición $0\rightarrow1\rightarrow2\rightarrow3\rightarrow0$... es más exacta que considerar las actividades de los bits de manera individual, en donde $a_{c(1)}=0{,}5$ y $a_{c(0)}=1$. Para realizar esto, habría que ampliar el concepto de actividad de una señal y el de las posibilidades de transición, pues ya no son las cuatro posibles transiciones de un bit ($0\rightarrow0$, $0\rightarrow1$, $1\rightarrow0$, $1\rightarrow1$), sino un gran número de transiciones diferentes que dependerán del ancho de la señal ($0\rightarrow0$, $0\rightarrow1$, $0\rightarrow2$, $0\rightarrow3$, $1\rightarrow0$, $1\rightarrow1$, $1\rightarrow2$,...). Posteriormente habría que trasladar las nuevas probabilidades de transición a la actividad a nivel de bits.

2. En lo referido a las **mejoras de la implementación**, se trata fundamentalmente de un trabajo de desarrollo. Por ser una primera propuesta, la implementación del método en una herramienta informática es preliminar. Como se ha comentado, el subconjunto del VHDL aceptado es reducido, por tanto, un primer paso sería ampliar el conjunto del VHDL a al menos el subconjunto del VHDL para síntesis.

   También sería conveniente ampliar el análisis a diseños jerárquicos. Hasta ahora, los diseños estructurales se *aplanaban* para que se pudiesen analizar por la herramienta. Ésta sería una mejora técnica, ya que no implica que se tenga que modificar el modelo propuesto.

   Como resultado de estas mejoras se podrían analizar de manera fácil circuitos de mayor tamaño. Esto permitiría ampliar los resultados experimentales y analizar el comportamiento del modelo propuesto en condiciones más desfavorables.

3. Un campo que requiere mayor investigación es el relativo a la **ampliación del método propuesto**. En éste hay varios aspectos importantes:
    - Consideración de los **elementos de memoria**. Para que el método resulte aplicable, es necesario incluir circuitos secuenciales. Dentro del nivel de puertas hay trabajos que consideran los elementos de memoria [46], [133], [120], [5], por tanto, una primera aproximación podría ser adaptar alguna de estas propuestas. El siguiente paso consistiría en investigar las características RTL que facilitan el estudio de los circuitos secuenciales. Un ejemplo de esto podría ser la extracción de las probabilidades y actividades de la señal que define el estado del circuito a partir del proceso VHDL que define su máquina de estados finitos. Para aprovechar al máximo este caso, se le podría unir la consideración de señales *multibit*.
    - Consideración de las **dependencias espaciales en las entradas**. Diversos trabajos previos han propuesto considerar las dependencias espaciales [80], [7], [5]. Desde el nivel RT, el uso de las señales multibit y la consideración de su actividad de manera conjunta, proporciona un medio más exacto para la consideración de las dependencias espaciales entre los bits de una misma señal. El uso de señales multibit ya se ha propuesto como línea futura en un punto anterior. Por otro lado, las dependencias espaciales entre distintas señales, en un principio se podrían considerar de manera similar a como se ha realizado en los anteriores trabajos, aunque con la ampliación de que las señales son multibit. Posteriormente, convendría investigar otras soluciones que aprovechen las características del nivel RT.
    - Estudio de las **transiciones espurias** (*glitches*). Las transiciones espurias aumentan considerablemente la actividad de un circuito. Estas transiciones dependen del retardo de las puertas y de la manera en que estén dispuestas en el circuito. Por tanto, esto son datos que desde el nivel RT no se puede conocer, ya que dependen del sintetizador, de la biblioteca utilizada y de las opciones de síntesis. Incluso, la estimación no resulta fácil de llevar a cabo desde el nivel de puertas [113]. Por tanto, desde el nivel RT no se puede aspirar a conocer la influencia de estas transiciones en la actividad, por lo que se obtiene un modelo de actividad sin retardos, que calcula la actividad mínima del circuito. Aún así, dentro de las limitaciones del nivel RT, se podría realizar una estimación gruesa de la influencia de las transiciones espurias según la profundidad combinacional de las señales y lo equilibrados que estén los caminos combinacionales.
4. La aplicación del método propuesto a otros análisis es un aspecto con un gran potencial. Algunas de las distintas líneas que se podrían desarrollar son:
    - **Estimación de consumo**: Ésta es la aplicación más evidente, ya que el consumo dinámico depende directamente de la actividad de conmutación. Existen herramientas como *XPower* [142] de *Xilinx* que estiman el consumo del circuito implementado en la FPGA a partir de los datos de actividad proporcionados por el usuario. En consecuencia, la obtención de estos datos de actividad constituiría una aplicación inmediata del método.

      También se podría realizar el cálculo del consumo dinámico estimando, a partir del EHM, la disposición de las puertas que constituirán el circuito. A partir de esta información y de la actividad de los nodos internos del circuito, se podría obtener una función del consumo dependiente de las capacidades internas de las puertas del circuito. Estas capacidades dependerán de la tecnología que se utilizará y, en un modelo más complejo, del *fanout* y características de cada puerta.

      Por último, ahora que el consumo estático está siendo cada vez más importante, el conocimiento de la actividad de conmutación puede ayudar a encontrar los compromisos entre el consumo dinámico y estático. El consumo estático puede ser estimado con la ayuda del EHM, ya que proporciona una estructura del circuito que ayudaría a estimar el número de puertas finales y su tamaño. Estos son parámetros fundamentales en el consumo estático. Por otro lado, debido a la dependencia del consumo estático con los valores lógicos de las entradas de las puertas [39], [25], la

obtención de las probabilidades de señal necesarias para el cálculo de la actividad también podrían contribuir a la estimación de este otro consumo.

- **Estimación de área**: El análisis realizado del circuito y la elaboración del EHM constituye un paso intermedio hacia la síntesis. Por tanto, la información del EHM junto con el análisis de los BDD puede servir para estimar el área del diseño. Resultados favorables [76] ya han sido obtenidos con el SHM (§3.3.1), por tanto, es de esperar que la ampliación del modelo permita obtener mejores resultados.
- **Análisis de testabilidad**: La actividad de una señal puede ser un indicativo de su controlabilidad. Detectar una señal con actividad muy baja puede revelar su baja controlabilidad, lo que se considera un parámetro importante en la testabilidad. Además, el análisis del EHM puede ayudar a detectar zonas poco observables.

  Por otro lado, los valores de actividad y probabilidad obtenidos pueden ayudar a identificar vectores de test poco adecuados y contribuir a encontrar vectores de test apropiados.
- **Análisis de fiabilidad**: los niveles altos de actividad pueden comprometer la fiabilidad de un circuito por las altas temperaturas alcanzadas debidas al consumo dinámico. Además, las altas actividades pueden generar un ruido que podría afectar a los circuitos mixtos (analógicos y digitales). El conocimiento de las actividades junto con el de la estructura del circuito proporcionado por el EHM podría contribuir a detectar zonas calientes y zonas donde no convendría tener señales digitales y analógicas cercanas.
- **Análisis funcional**: Los BDD aportan información sobre la funcionalidad de las señales, de hecho, a partir del BDD de probabilidad se puede obtener la función lógica de la señal correspondiente. Por tanto, el análisis de los BDD de las señales de un circuito, junto con el conocimiento de su estructura aportado por el EHM, podrían contribuir a un análisis cuyo objetivo sea el conocimiento de la funcionalidad del circuito.
- **Análisis de calidad**. La ampliación del SHM, dando como resultado el EHM, junto con el análisis funcional aportado por los BDD, proporciona un mayor número de posibilidades de análisis de calidad. Muchos de estos análisis resultan inmediatos de implementar y, de hecho, han salido a la luz durante la fase experimental de esta tesis, en donde se han detectado muchos casos de los que se enumeran a continuación. Estos estudios contribuirían a ampliar la línea de investigación relativa al análisis de calidad llevada a cabo en el departamento [20], [130], [132]. Ejemplos de estos análisis son:
  - Detección de condiciones de estructuras *if* a las que nunca se llega.
  - Detección de condiciones imposibles (nunca se cumplen) y tautologías (condiciones que siempre se cumplen).
  - Detección de código VHDL inútil, es decir, código que se simplifica por una constante, por una señal o por una expresión más sencilla que la original.
  - Detección que señales que aunque figuren como influencias de otra señal, realmente no influyen. Esto puede ser para dependencias inmediatas, como para dependencias que no influyan directamente sobre la señal.
- **Verificación funcional**: Los valores de probabilidad de las señales de un circuito en cierta medida reflejan su funcionalidad. Esto se debe a que la ecuación de probabilidad de una señal es idéntica a su función lógica con la única salvedad de que, en vez de asignar valores binarios ($x_i \in \{0, 1\}$) a las variables x_i de la función, se asignan probabilidades ($x_i \in [0, 1]$). Esto es, el dominio de la función pasa de ser discreto a continuo. Como resultado, a la imagen de la función[55] le sucederá lo mismo:

[55] Los valores que puede tomar la función

al asignar un dominio continuo, la imagen también pasa a ser continua y con el mismo rango de valores $f \in [0, 1]$; en vez de discretos ($f \in \{0,1\}$).

Asignar valores de probabilidad, esto es valores continuos, sería similar a realizar una simulación de tiempo infinito en la que los vectores de entrada tienen las probabilidades asignadas y que como resultado se obtiene la relación del tiempo que la señal está a nivel alto sobre el tiempo total. Por ser un análisis probabilístico y no estadístico, el tiempo total es infinito y por tanto, el resultado no depende de los valores concretos de los vectores de test empleados en la simulación. Este planteamiento es similar a la distinción entre los métodos estáticos (§2.1.2) y dinámicos (§2.1.1) explicados en esta tesis pero aplicados a la funcionalidad en vez de a la actividad y probabilidad.

Si en las entradas no se asignan unas probabilidades extremas (muy cercanas a 0 ó a 1), es muy difícil que dos circuitos similares aunque no idénticos tengan entre ellos igual probabilidad en sus señales internas y salidas.

Esto se ha podido comprobar durante la fase experimental de esta tesis doctoral. Como se comentó en el capítulo de resultados experimentales, los circuitos RTL se han tenido que adaptar manualmente para poder ser analizados automáticamente por la herramienta[56]. Entre estas adaptaciones figuraban el aplanado del circuito y la sustitución de señales multibit por señales de un solo bit. Como es habitual en todo procedimiento manual, era habitual que se incurriese en algún error durante adaptaciones, como por ejemplo lo sería asignar un índice de un vector diferente u olvidar el actualizar el nombre de alguna señal después de copiar y pegar. Para asegurar que la adaptación se había realizado correctamente, se comprobaba mediante simulación la exactitud entre el circuito RTL adaptado con el circuito sintetizado en puertas.

Durante las comprobaciones se observó que siempre que la simulación indicaba la existencia de error en alguna señal, los valores de probabilidad de dicha señal proporcionados por los modelos RTL y puertas diferían. Y de manera similar ocurría con lo contrario: si las simulaciones mostraban resultados idénticos, también lo eran los valores de probabilidad proporcionados por ambos modelos. Esto se comprobó para todos los circuitos y en ningún caso sucedió que con valores iguales de probabilidad saliesen valores distintos en simulación, ni tampoco lo contrario. Evidentemente, esto requiere un estudio más profundo, ya que sólo se ha comprobado para los ejemplos propuestos en esta tesis. Pero hace augurar una posible **aplicación del método propuesto en la verificación funcional**.

El método podría ser ampliado. Por ejemplo, para asegurar que dos circuitos son equivalentes se puede comprobar con distintos valores de probabilidad a las entradas (lo que se correspondería con varias simulaciones). Además, en el caso de que no se hiciesen particiones disjuntas, los valores de actividad también podrían ser comparados, ya que también serían exactos. Esto aportaría más seguridad en la verificación, aunque no parece que sea necesaria tanta seguridad. Incluso si se quisiese llegar más lejos, se podrían obtener los valores de probabilidad y actividad para toda la malla de valores propuesta en la figura 4-10 y obtener una especie de firma del circuito con sus valores de probabilidad y actividad para todo la malla.

Los beneficios de la técnica probabilística son evidentes, ya que es mucho más rápido y cómodo realizar un análisis automático probabilístico que realizar una simulación, en donde hay que crear el diseño del banco de pruebas y los vectores de test, y hay que esperar los largos tiempos de simulación. Además, los resultados del método probabilístico son más fáciles de comprobar, ya que se comparan números y no formas de onda. Por último, la búsqueda de la zona del circuito donde radican las

[56] Los circuitos en puertas no se tuvieron que adaptar porque la síntesis puede producir circuitos "aplanados" y con señales de un sólo bit

diferencias puede ser asistida por el EHM, ya que se conocen las señales internas que tienen probabilidades diferentes, y el EHM proporciona el árbol de dependencias y retroalimentación con el código. Así que el EHM podría indicar dónde se produce la raíz del error: primera señal que no tiene probabilidades iguales y cómo se propagan esas diferencias.

REFERENCIAS

[1] Agarwal A., Mukhopadhyay S., Kim C.H., Raychowdhury A., Roy K., *Leakage Power Analysis and Reduction: Models, Estimation and Tools*, IEE Proceedings - Computers and Digital Techniques, vol. 152, nº 3, mayo 2005

[2] Agrawal V., Seth S., *Mutually Disjoint Signals and Probability Calculation in Digital Circuits*, Great Lakes Symposium on VLSI, Lafayette, Luisiana, EE.UU., febrero 1998

[3] Assaderaghi F., Sinitsky D., Parke S., Bokor J., Ko P., Hu C., *Dynamic Threshold-Voltage MOSFET (DTMOS) for Ultra-Low Voltage VLSI*, IEEE Transactions on Electron Devices, vol. 44, nº 3, marzo 1997

[4] Benkoski J., *The Metrics of Success*, Chip Design Magazine, junio-julio 2004

[5] Bhanja S., Lingasubramanian K., Ranganathan N., *Estimation of Switching Activity in Sequential Circuits using Dynamic Bayesian Networks*, International Conference on VLSI Design, Calcuta, India, enero 2005

[6] Bhanja S., Ranganathan N., *Dependency Preserving Probabilistic Modeling of Switching Activity using Bayesian Networks*, Design Automation Conference, Las Vegas, Nevada, EE.UU., junio 2001

[7] Bhanja S., Ranganathan N., *Modeling Switching Activity Using Cascaded Bayesian Networks for Correlated Input Streams*, International Conference on Computer Design, Friburgo, Alemania, septiembre 2002

[8] Bhanja S., Ranganathan N., *Switching Activity Estimation of Large Circuits using Multiple Bayesian Networks*, International Conference on VLSI Design, Bangalore, India, enero 2002

[9] Bennett C., *Notes on the History of Reversible Computation*, IBM Journal of Research and Development, vol. 32, nº1, enero 1988

[10] Bennett C., *Notes on Landauer's Principle, Reversible Computation, and Maxwell's Demon*, Studies In History and Philosophy of Modern Physics, vol. 34 nº 3, septiembre 2003. arXiv:physics/0210005

[11] Bennett P., *The Why, Where and What of Low-Power SoC Design*, revista electrónica EETimes, 12 mayo 2004, http://www.eetimes.com/showArticle.jhtml?articleID=54202129

[12] Bernacchia G., Papaefthymiou M., *Analytical Macromodeling for High-Level Power Estimation*, International Conference on Computer Aided Design, San José, California, EE.UU., noviembre 1999

[13] Bollig B., Wegener I., *Improving the Variable Ordering of OBDDs is NP-complete*, IEEE Transactions on Computers, vol. 45, no.9, septiembre 1996

[14] Bowyer B., *The 'What' and 'Why' of Transaction Level Modeling*, revista electrónica EETimes, 27 febrero 2006, http://www.eetimes.com/showArticle.jhtml?articleID=180207854

[15] Brglez F., Fujiwara H., *A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran*, International Symposium of Circuits and Systems, Kyoto, Japón, junio 1985

[16] Bryant R., *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Transactions on Computers, vol. 35 no. 8, agosto 1986

[17] Bryant R., *Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams*, ACM Computing Surveys, vol. 24 nº 3, septiembre 1992

[18] Burch R., Najm F., Yang P., Trick T., *A Monte Carlo Approach for Power Estimation*, IEEE Transactions on Very Large Scale Integration Systems, vol. 1, nº 1, marzo 1993

[19] Cadence, *Learning to Live with Electromigration*, White Paper, Cadence Design Systems, Inc, 2002. http://www.cadence.com/whitepapers/4252_EM_WP_fnl.pdf

[20] Casado F., Machado F., Martínez N., Torroja Y., Neumann P., Seepold R., *Study of Different Kind of Tools to Analyse the Quality of HDL Designs. Comparison of their Coverage of the RMM Guideline*. Conference on Design of Circuits and Integrated Systems, Porto, Portugal, noviembre 2001

[21] Casado F., Riesgo T., Torroja Y., Santos M., Entrena L., *Comparison of Different Testability Analysis Techniques for Pre-synthesized Descriptions*, Conference on Design of Circuits and Integrated Systems, Ciudad Real, España, noviembre 2003

[22] Chandrakasan A., Brodersen R., *Minimizing Power Consumption in digital CMOS Circuits*, Proceedings of the IEEE, vol. 83 nº 4, abril 1995

[23] Chen C., Ahmadi M., *Register-Transfer-Level Power Estimation Based On Technology Decomposition*, IEE Proceedings - Circuits, Devices and Systems, vol. 151, nº 2, abril 2004

[24] Chen Z., Diaz C. H., Plummer J., Cao M., Greene W., *0.18 um dual Vt MOSFET Process and Energy-Delay Measurement*, IEEE Electron Devices Meeting, San Francisco, California, EE.UU., diciembre 1996

[25] Chen Z., Johnson M., Wei L., Roy K., *Estimation of Standby Leakage Power in CMOS Circuit Considering Accurate Modeling of Transistor Stacks*, International Symposium on Low Power Electronics and Design, Monterey, California, EE.UU., agosto 1998

[26] Chou T., Roy K., *Estimation of Sequential Circuit Activity Considering Spatial and Temporal Correlations*, International Conference on Computer Design, Austin, Texas, EE.UU., octubre 1995

[27] Chou T., Roy K., *Statistical Estimation of Sequential Circuit Activity*, International Conference on Computer Aided Design, San José, California, EE.UU., noviembre 1995

[28] Chou T., Roy K., Prasad S., *Estimation of Circuit Activity Considering Signal Correlations and Simultaneous Switching*, International Conference on Computer Aided Design, San José, California, EE.UU., noviembre 1994

[29] Cirit M., *Estimating Dynamic Power Consumption of CMOS Circuits*, International Conference on Computer Aided Design, Santa Clara, California, EE.UU., noviembre 1987

[30] Claasen T., *The Logarithmic Law of Usefulness*, Semiconductor International, julio 1998

[31] Costa J., Monteiro J., Devadas S., *Switching Activity Estimation using Limited Depth Reconvergent Path Analysis*, International Symposium on Low Power Electronics and Design, Monterey, California, EE.UU., agosto 1997

[32] Cygwin, http://www.cygwin.com

[33] Dennard R., Gaensslen F., Yu H-N, Riedout L., Bassous E., Leblanc A., *Design Of Ion-implanted MOSFET's with Very Small Physical Dimensions*, IEEE Journal of Solid-State Circuits, vol. 9, nº 5, octubre 1974

[34] Ding C-S, Wu Q., Hsieh C-T, Pedram M., *Stratified Random Sampling for Power Estimation*, IEEE Transactions on Computer Aided-Design of Integrated Circuits and Systems, vol. 17, nº 6, junio 1998

[35] Dresig F., Lanches P., Rettig O., Baitinger U., *Simulation and Reduction of CMOS Power Dissipation at Logic Level*, European Conference on Design Automation, París, Francia, febrero 1993

[36] Durrani Y., Riesgo T., Machado F., *Statistical Power Estimation for Register Transfer Level*, Mixed Design of Integrated Circuits and Systems Conference, Gdynia, Polonia, junio 2006

[37] Ercolani S., Favalli M., Damiani M., Olivo P., Riccò B., *Estimate of Signal Probability in Combinational Logic Networks*, European Test Conference, París, Francia, abril 1989

[38] Fernandes J., Santos M., Oliveira A., Teixeira J., *A Probabilistic Method for the Computation of Testability of RTL Constructs*, Design, Automation and Test in Europe Conference, París, Francia, febrero 2004

[39] Ferré A., Figueras J., *On Estimating Leakage Power Consumption for Submicron CMOS Digital Circuits*, International Workshop on Power And Timing Modeling, Optimization and Simulation, Lovaina la Nueva, Bélgica, septiembre 1997

[40] Ferré A., Figueras J., *Leakage Power Bounds in CMOS Digital Technologies*, IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems, vol. 21, nº 6, junio 2002

[41] Ferreira R., Trullemans A., Costa J., Monteiro J., *Probabilistic bottom-up RTL power estimation*, International Symposium on Quality Electronic Design, San José, California, marzo 2000

[42] Figueras J., *Low Power Circuit and Logic Level Design, Modeling*, en [95], sección 4.1, páginas 81-104

[43] Flynn M., Hung P., *Microprocessor Design Issues: Thoughts on the Road Ahead*, IEEE Micro, vol. 25, nº 3, mayo-junio 2005
[44] Frank D., *Power-Constrained CMOS Scaling Limits*, IBM Journal of Research and Development, vol. 46, nº 2/3, marzo-mayo 2002
[45] Garey M., Johnson D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W H Freeman & Co. junio 1979
[46] Ghosh A., Devadas S., Keutzer K., White J., *Estimation of Average Switching Activity in Combinational and Sequential Circuits*, Design Automation Conference, Anaheim, California, EE.UU., junio 1992
[47] Goering R., *Productivity may stumble at 100nm*, revista electrónica EETimes, 23 septiembre 2003, http://www.eetimes.com/story/OEG20030923S0041
[48] Graphviz, http://www.graphviz.org/
[49] Gupta S., Najm F., *Analytical Models for RTL Power Estimation of Combinational and Sequential Circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems, vol. 19, nº 7, junio 2000
[50] Häggström O., *Finite Markov Chains and Algorithmic Applications*, Cambridge University Press, 2002
[51] Halter J., Najm F., *A Gate-Level Leakage Power Reduction Method for Ultra-Low-Power CMOS Circuits*, Custom Integrated Circuits Conference, Santa Clara, California, EE.UU., mayo 1997
[52] Harlow J., Brglez F., *Design of Experiments and Evaluation of BDD Ordering Heuristics*, International Journal on Software Tools for Technology Transfer, vol. 3, no. 2, mayo 2001
[53] Hansen M., Yalcin H., Hayes J., *Unveiling the ISCAS-85 Bench*marks*: A Case Study in Reverse Engineering*, IEEE Design and Test, vol. 16 no. 3, julio-septiembre 1999
[54] Hemani A., *Charting the EDA Roadmap*, IEEE Circuits & Devices Magazine, vol. 20, nº 6, noviembre-diciembre 2004
[55] Hofstee P., *Future Microprocessors and Off-Chip SOP Interconnect*, IEEE Transactions on Advanced Packaging, vol. 27, nº 2, mayo 2004
[56] Horowitz M., Alon E., Patil D., Naffziger S., Kumar R., *Scaling, Power, and the Future of CMOS*, IEEE International Electron Devices Meeting, Washington DC, EE.UU., diciembre 2005
[57] Hu F., Agrawal V., *Enhanced Dual-Transition Probabilistic Power Estimation with Selective Supergate Analysis*, International Conference on Computer Design, San José, California, EE.UU., octubre 2005
[58] Hutchby J., Shirnov V., Cavin R., Bourianoff G., *Functional Scaling beyond Ultimate CMOS*, International Conference on Solid-State and Integrated-Circuit Technology, Beijing, China, octubre 2004
[59] IEEE, Institute of Electrical and Electronic Engineers, *IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1993*, 1994
[60] IEEE, Institute of Electrical and Electronic Engineers, *IEEE Standard for VHDL Register Transfer Level (RTL) Synthesis, IEEE Std 1076.6-2004,* 2004
[61] IBS, International Business Strategies, *Analysis of the Relationship between EDA Expenditures and Competitive Positioning of IC Vendors*, Los Gatos, California, 2002
[62] IBS, International Business Strategies, *Analysis of the Relationship between EDA Expenditures and Competitive Positioning of IC Vendors for 2003*, Los Gatos, California, 2003
[63] ITRS, *International Technology Roadmap for Semiconductors*, http://public.itrs.net, Edición 2007
[64] Lind-Nielsen J., *BuDDy: Binary Decision Diagram Package*, noviembre 2002, http://buddy.sourceforge.net
[65] Jones H., *Alleviating Design Schedule Misses*, Electronics Design Chain, vol. 2, primavera 2003
[66] Kawa J., MacMillen D., *Controlling Power Is Paramount to Design*, Integrated System Design, nº 13150, diciembre 2001

[67] Kim N., Austin T., Blaauw D., Mudge T., Flautner K., Hu J., Irwin M., Kandemir M., Narayanan V., *Leakage Current: Moore's Law Meets Static Power*, Computer, vol. 36, no. 12, diciembre 2003

[68] Kish L., *Moore's Law and the Energy Requirement of Computing versus Performance*, IEE Proceedings - Noise in Devices and Circuits, vol. 151, nº 2, abril 2004

[69] Krieger R., *PLATO: A Tool for Computation of Exact Signal Probabilities*, *International Conference on VLSI Design*, Mumbai, India, enero 1993

[70] Krishnamoorthy A., *Minimize IC Power without Sacrificing Performance*, revista electrónica EETimes, 15 julio 2004, http://www.eetimes.com/showArticle.jhtml?articleID=23901143

[71] Krishnamurthy B., Tollis I., *Improved Techniques for Estimating Signal Probabilities*, IEEE Transactions on Computers, vol. 38, nº 7, julio 1989

[72] Kuroda T., Fujita T., Mita S., Nagamatsu T., Yoshioka S., Suzuki K., Sano F., Norishima M., Murota M., Kako M., Kinugawa M., Kakumu M., Sakurai T., *A 0.9V 150MHz 10mW 4mm2 2-D Discrete Cosine Transform Core Processor with Variable-Threshold-Voltage Scheme*, IEEE Journal of Solid State Circuits, vol. 31 nº 11, noviembre 1996

[73] Lam W., Brayton R., Sangiovanni-Vincentelli A., *Valid Clocking in Wavepipelined Circuits*, International Conference on Computer Aided Design, Santa Clara, California, EE.UU., noviembre 1992

[74] Landauer R., *Dissipation and Heat Generation in the Computing Process*, IBM Journal of Research and Development, vol. 5 nº3, julio 1961

[75] Lefurgy C., Rajamani K., Rawson F., Felter W., Kistler M., Keller T., *Energy Management for Commercial Servers*, Computer, vol. 36, no. 12, diciembre 2003

[76] Machado F., Torroja Y., Casado F., Riesgo T., Torre E., Uceda J., *A Simple Method to Estimate the Area of VHDL RTL descriptions*, Conference on Design of Circuits and Integrated Systems, Montpellier, Francia, noviembre 2000

[77] Macii A., Macii E., Poncino M., Scarsi R., *Stream Synthesis for Efficient Power Simulation based on Spectral Transforms*, IEEE Transactions on Very Large Scale Integration Systems, vol. 9, nº 3, junio 2001

[78] Malik S., Wang A., Brayton R., Sangiovanni-Vincentelli A., *Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment*, International Conference on Computer Aided Design, Santa Clara, California, EE.UU., noviembre 1988

[79] Marculescu D., *Information Theoretic and Probabilistic Measures for Power Analysis of Digital Circuits*, Tesis Doctoral, University of Southern California, agosto 1998

[80] Marculescu R., Marculescu D., Pedram M., *Switching Activity Analysis Considering Spatiotemporal Correlations*, International Conference on Computer Aided Design, San José, California, EE.UU., noviembre 1994

[81] Marculescu R., Marculescu D., Pedram M., *Sequence Compaction for Power Estimation: Theory and Practice*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, nº 7, julio 1999

[82] Marković D., Stojanović V., Nikolić B., Horowitz M., Brodersen R., *Methods for True Energy-Performance Optimization*, IEEE Journal of Solid-State Circuits, vol. 39, nº 8, agosto 2004

[83] Markowsky G., *Bounding Signal Probabilities in Combinatorial Circuits*, IEEE Transactions on Computers, vol. 36, nº 10, octubre 1987

[84] Méndez M., González J., Mateo D., A. Rubio, *An Investigation on the Relation between Digital Circuitry Characteristics and Power Supply Noise Spectrum in Mixed-Signal CMOS Integrated Circuits*, Microelectronics Journal, vol. 36, nº 1, enero 2005

[85] Mentor Graphics, Modelsim, http://www.model.com/, Mentor Graphics

[86] Moore G., *Cramming More Components onto Integrated Circuits*, Electronics, vol. 38, nº 8, 19 abril 1965

[87] Moore G., *No exponential is forever: but "Forever" can be delayed!*, IEEE International Solid-State Circuits Conference, San Francisco, CA, EE.UU., febrero 2003

[88] Mudge T., *Power: a First-Class Architectural Design Constraint*, Computer, vol. 34, no. 4, abril 2001

[89] Mutoh S., Douseki T., Matsuya Y., Aoki T., Shigematsu S., Yamada J., *1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS,* IEEE Journal of Solid State Circuits, vol. 30 nº 8, agosto 1995

[90] Najm F., *Transition Density, A Stochastic Measure of Activity in Digital Circuits*, Design Automation Conference, San Francisco, California, EE.UU., junio 1991

[91] Najm F., *A Survey of Power Estimation Techniques in VLSI Circuits*, IEEE Transactions on Very Large Scale Integration Systems, vol. 2 nº 4, diciembre 1994

[92] Najm F., Abraham J., *Accounting for Very Deep Sub-Micron Effects in Silicon Models,* revista electrónica EETimes, 9 enero 2001., http://www.eetimes.com/showArticle.jhtml?articleID=17406932

[93] Najm F., Burch R., Yang P., Hajj I., *CREST - A Current Estimator for CMOS Circuits*, International Conference on Computer Aided Design, Santa Clara, California, EE.UU., noviembre 1988

[94] Nardi A., Zeng H., Garret J., Daniel L., Sangiovanni-Vicentelli A., *A Methodology for the Computation of an Upper Bound on Noise Current Spectrum of CMOS Switching Activity*, International Conference on Computer Aided Design, San José, California, EE.UU., noviembre 2003

[95] Nebel W., Mermet J., editores, *Low Power Design in Deep Submicron Electronics*, Kluwer Academic Publishers, 1997

[96] Nebel W., *Power Estimation at the Logic Level*, en [95], sección 4.3

[97] Nebel W., *Predictable Design of Low Power Systems by Pre-Implementation Estimation and Optimization*, Asian South Pacific Design Automation Conference (ASP-DAC), Yokohama, Japón, enero 2004

[98] Nemani M., Najm F., *Towards a High-Level Power Estimation Capability*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, nº 6, junio 1996

[99] Nose K., Sakurai T., *Analysis and Future Trend of Short-Circuit Power*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, nº 9, septiembre 2000

[100] Nourivand A., Wang C., Ahmad O., *A VHDL-based technique for an accurate estimation of leakage power in digital CMOS circuits*, International IEEE-NEWCAS Conference, Quebec, Canadá, junio 2005

[101] Oregano, http://www.oregano.at/ip/8051.htm, Oregano Systems

[102] Papadimitriou C., *Computational Complexity*, Addison-Wesley Publishers Company, noviembre 1993

[103] Parker K., McCluskey E., *Analysis of Logic Circuits with Faults using Input Signal Probabilities*, International Symposium on Fault Tolerant Computing, Champaign, Illinois, EE.UU., junio 1974

[104] Parra P., Acosta A., Valencia M., *Reduction of Switching Noise in Digital CMOS Circuits for Pin Swapping of Library Cells,* International Workshop on Power and Timing Modeling, Optimization and Simulation, Yverdon-les-bains, Suiza, septiembre 2001

[105] Patel C., Sharma R., Bash C., Graupner S., *Energy Aware Grid: Global Workload Placement based on Energy Efficiency*, Hewlett-Packard Laboratories Technical Report, HPL-2002-329, 21 noviembre 2002

[106] Rabaey J. M., Chandrakasan A., Nikolic B., *Digital Integrated Circuits, A Design Perspective*, Segunda Edición, Editorial Prentice Hall, 2002

[107] Raja T., Agrawal V., Bushnell M., *Variable Input Delay CMOS Logic for Low Power Design*, International Conference on VLSI Design, Calcuta, India, enero 2005

[108] Real Academia Española, *Diccionario de la lengua Española*, XXII edición, 2001

[109] Ross D., *Power Struggle [Power Supplies for Portable Equipment]*, IEE Review, vol. 49, nº 7, julio 2003

[110] Ross P., *Beat the Heat*, IEEE Spectrum, vol. 41, no. 5, mayo 2004

[111] Roy K., Prasad S., *Low-Power CMOS VLSI Circuit Design*, Editorial John Willey & Sons, 2000

[112] Rudell R., *Dynamic Variable Ordering for Ordered Binary Decision Diagrams*, International Conference on Computer Aided Design, Santa Clara, California, EE.UU., noviembre 1993

[113] Ruiz de Clavijo P., Bellido M., Juan J., *HALOTIS - High Accurate Logic Timing Simulator*, PhD Research in Microelectronics and Electronics (PRIME), Lausana, Suiza, julio 2005
[114] Sangiovanni-Vincentelli A., *The Tides of EDA*, IEEE Design & Test of Computers, vol. 20, nº 6, noviembre-diciembre 2003
[115] Sangiovanni-Vincentelli A., *System Level Design: a Strategic Investment for the Future of the Electronic Industry*, International Symposium on VLSI Design, Automation and Test, Hsinchu, Taiwán, abril 2005
[116] Santarini M., *Power is a Growing Concern at 90, 65 nm*, revista electrónica EETimes, 5 febrero 2004, http://www.eetimes.com/showArticle.jhtml?articleID=17601961
[117] Schneider P., Senn M., Wurth B., *Power Analysis for Sequential Circuits at Logic Level*, European Design Automation Conference, Ginebra, Suiza, septiembre 1996
[118] Schneider P., Schlichtmann U., Antreich K., *A New Power Estimation Technique with Application to Decomposition of Boolean Functions for Low Power,* European Design Automation Conference, Grenoble, Francia, septiembre 1994
[119] Schneider P., Schlichtmann U., Wurth B., *Fast Power Estimation of Large Circuits*, IEEE Design & Test of Computers, vol. 12, no. 1, primavera 1996
[120] Schneider P., Wurth B., *Transition Probability Estimation for Combinational and Sequential Circuits*, International Workshop on Logic Synthesis, Lake Tahoe, California, EE.UU., mayo 1995
[121] Seth S., Pan L., Agrawal V., *PREDICT - Probabilistic Estimation of Digital Circuit Testability*, International Fault Tolerant Computing Symposium, Ann Arbor, Michigan, junio 1985
[122] Seth S., Agrawal V., *A New Model for Computation of Probabilistic Testability in Combinational Circuits*, Integration, the VLSI Journal, vol. 7, nº 1, abril 1989
[123] Shannon C., *The Mathematical Theory of Communication*, Bell System Technical Journal, vol. 27, julio-octubre 1948
[124] Smith T., *Gelsinger: 100W Power Dissipation 'OK' for Desktops*, The Register, 15 septiembre 2003, http://www.theregister.co.uk/2003/09/15/gelsinger_100w_power_dissipation_ok
[125] Synopsys, *Design Compiler™ User Guide*, Synopsys
[126] Synopsys, *Power Compiler™ User Guide*, Synopsys
[127] The Economist, *Why Speed isn't Everything*, The Economist, Technology Quarterly, 13 mayo 2004
[128] Theodoridis G., Goutis C., *Logic Level Power Estimation*, capítulo 9 del libro, Designing CMOS Circuits for Low Power, Editores: D. Soudris, C. Piguet, C. Goutis, Kluwer Ac. Pub. 2002
[129] Theodoridis G., Theoharis S., Soudris D., Goutis C., *Switching Activity Estimation under Real-Gate Delay Using Timed Boolean Functions*, IEE Proceedings - Computers and Digital Techniques, vol. 147, nº 6, noviembre 2000
[130] Torroja Y., *Estudio de la Calidad de Circuitos Electrónicos Descritos en VHDL. Propuesta de un Conjunto de Métricas*, Tesis Doctoral, Departamento de Automática, Ingeniería Electrónica e Informática Industrial, Universidad Politécnica de Madrid, España, 2000
[131] Torroja Y., Casado F., Machado F., Riesgo T., Torre E., Uceda J., *Using a Simplified Hardware Model to Analyse the Quality of VHDL Based Designs*, Design Automation and Test in Europe (User Forum), París, Francia, marzo 2000
[132] Torroja Y., Machado F., Casado F., Torre E., Riesgo T., Uceda J., *ARDID: A Tool and a Model for the Quality Analysis of VHDL based Design*, en el libro *Virtual Components Design and Reuse,* R. Seepold, N. Martínez eds., Kluwer Academic Publishers, 2001
[133] Tsui C., Monteiro J., Pedram M., Devadas S., Despain A., Lin B., *Power Estimation Methods for Sequential Circuits*, IEEE Transactions on Very Large Scale Integration Systems, vol. 3 no. 3, septiembre 1995
[134] Tsui C., Pedram M., Despain A., *Efficient Estimation of Dynamic Power Consumption under a Real Delay Model*, International Conference on Computer Aided Design, Santa Clara, California, EE.UU., noviembre 1993
[135] Usami K., Igarashi M., *Low-power Design Methodology and Applications utilizing Dual Supply Voltages*, Asia South Pacific Design Automation Conference, Yokohama, Japón, enero 2000

[136] von Neumann J., *Theory of Self-Reproducing Automata*, editado y completado por A. W. Burks, University of Illinois Press, Urbana y Londres 1966
[137] Wei L., Roy K., De V. K., *Low Voltage Low Power CMOS Design Techniques for Deep Submicron ICs*, International Conference on VLSI Design, Calcuta, India, enero 2000
[138] Wright R., Shanblatt M., *Reducing BDD Size by Exploiting Structural Connectivity*, Great Lakes Symposium on VLSI, Ypsilanti, Michigan, EE.UU., marzo 1999
[139] Wright R., Shanblatt M., *Improved Power Estimation for Behavioral and Gate Level Designs*, IEEE Computer Society Workshop on VLSI, Orlando, Florida, EE.UU., marzo 2001
[140] Wunderlich H-J., *PROTEST: A Tool for Probabilistic Testability Analysis*, Design Automation Conference, Las Vegas, Nevada, EE.UU., junio 1985
[141] Xakellis M., Najm F., *Statistical Estimation of the Switching Activity in Digital Circuits*, Design Automation Conference, San Diego, California, EE.UU., junio 1994
[142] Xilinx, *XPower*, http://www.xilinx.com
[143] Yang A., *Move beyond the Legacy of IR Drop*, revista electrónica EETimes, 8 septiembre 2003, http://www.eetimes.com/showArticle.jhtml?articleID=17408675
[144] http://www.vhdl.org/rassp/vhdl/models/MSI/synth_models

ANEXO I. ANEXOS RELATIVOS A LOS DIAGRAMAS DE DECISIÓN BINARIA

En este anexo se añade información adicional sobre los diagramas de decisión binaria.

AI.1 Algoritmos para recorrer un BDD

En el apartado 2.2.2 se introdujeron los BDD. En la figura 2-10 de este apartado se mostró cómo se recorre un BDD. A continuación se incluye la estructura de los algoritmos para recorrer un BDD.

El recorrido de un BDD se realiza partiendo de su nodo raíz. Aunque cada camino empieza desde el nodo raíz, se utiliza un algoritmo con llamadas recurrentes a la función que lo atraviesa y así se consiguen acortar los recorridos de los caminos. En este algoritmo se va guardando el camino transitado desde el nodo raíz, y se va actualizando al entrar y salir de cada nodo. Este algoritmo es bien conocido en la literatura de grafos, en el cual al llegar a cada nodo se vuelve a llamar a la función para cada uno de sus hijos. En el código AI-1 se muestra la estructura de esta función recurrente, con la llamada a la función se anota el nodo, se vuelve a llamar a la misma función para cada uno de sus hijos. Antes de salir de la función se borra del nodo del camino recorrido.

```
recorre_diagrama(nodo_raiz, camino_recorrido) {
  anade_al_camino_recorrido (nodo_raiz, camino_recorrido);
  hijo_cero = obten_nodo_rama_cero(nodo_raiz);
  hijo_uno  = obten_nodo_rama_uno(nodo_raiz);
  recorre_diagrama(hijo_cero);
  recorre_diagrama(hijo_uno);
  borra_del_camino_recorrido (nodo_raiz, camino_recorrido);
}
```

Código AI-1: Función recurrente para recorrer un BDD

Sin embargo, para los BDD es interesante conocer qué rama se ha escogido para cada nodo, por lo que el algoritmo quedaría como el mostrado por el código AI-2. En este algoritmo, la función que añade el nodo al camino recorrido se llama dos veces, ya que su tercer argumento indica el tipo de rama que se ha escogido para el nodo. De la misma manera, la función que borra el camino también se llama dos veces.

```
recorre_diagrama_2(nodoraiz, camino_recorrido) {
  /* Rama cero */
  hijo_cero = obten_nodo_rama_cero(nodo_raiz);
  anade_al_camino_recorrido (nodoraiz, camino_recorrido,0);
  recorre_diagrama(hijo_cero);
  borra_del_camino_recorrido (nodo_raiz, camino_recorrido);
  /* Rama uno */
  hijo_uno  = obten_nodo_rama_uno(nodo_raiz);
  anade_al_camino_recorrido (nodoraiz, camino_recorrido,1);
  recorre_diagrama(hijo_uno);
  borra_del_camino_recorrido (nodo_raiz, camino_recorrido);
}
```

Código AI-2: Función recurrente para recorrer un BDD que recuerda las ramas transitadas

En la figura 2-10 se muestra un ejemplo del recorrido de un BDD.

En la sección AI.2 se muestra una extensión de este método para el cálculo de probabilidades y actividades. Con este método se consigue un ahorro muy significativo de tiempo de procesamiento para BDD de gran tamaño.

AI.2 Cálculo de probabilidades y actividades a partir del BDD

El cálculo de las probabilidades y las actividades de una señal a partir de su BDD se ha integrado en el paquete BuDDy [64] creando funciones específicas para ello. En esta tesis se ha explicado cómo se obtiene la ecuación de probabilidad y actividad de una señal a partir de su BDD:

- BDD de probabilidad: en la sección 2.2.2 figura 2-9
- TFBDD: sección 3.5.1 figura 3-34
- *a*BDD: sección 3.5.2.5 figura 3-62

Estos métodos son adecuados para la obtención de las ecuaciones de probabilidad y actividad, y para explicar el funcionamiento de los BDD. Sin embargo, estos métodos son computacionalmente caros ya que implican recorrer todo el BDD como si fuera un árbol y no hacen uso de las reducciones de nodos que se realizan en los BDD. El algoritmo utilizado para recorrer un BDD se esbozó en el código AI-2 y en la figura 2-10.

Para ayudar a entender este concepto se procederá a calcular la actividad de una XOR de tres entradas, cuyo BDD aparece en la figura 3-70. Siguiendo el método que ya se ha explicado, se atraviesan todos sus caminos del BDD contabilizándose aquellos que terminan en el nodo final uno. En realidad, como se explicó en la sección 2.2.2 (figura 2-10), no se empieza cada camino desde el nodo raíz, sino que se hace de forma recurrente considerando el camino recorrido, aún así en la figura se mostrarán los caminos como si partiesen desde el nodo raíz para facilitar la comprensión de las figuras.

Para este ejemplo existen cuatro caminos que terminan en el nodo final uno. En la figura AI-1 se ha mostrado el recorrido de todos los caminos. Para este caso, los caminos etiquetados en la figura con un número par son los que terminan en el nodo final uno. Son estos caminos los que se contabilizarán para la ecuación de probabilidad. Como se puede apreciar en la figura, con cada nodo que se avanza hacia abajo se añade un término a la ecuación del camino. Al llegar al nodo final se obtiene la ecuación de ese camino. En caso de que el camino termine en un nodo final uno, ese camino se añade al de la ecuación final, mientras que si se trata del nodo final cero no se tiene en cuenta. Como las ecuaciones se obtienen al descender por el camino, se ha denominado *cálculo descendente* a este método.

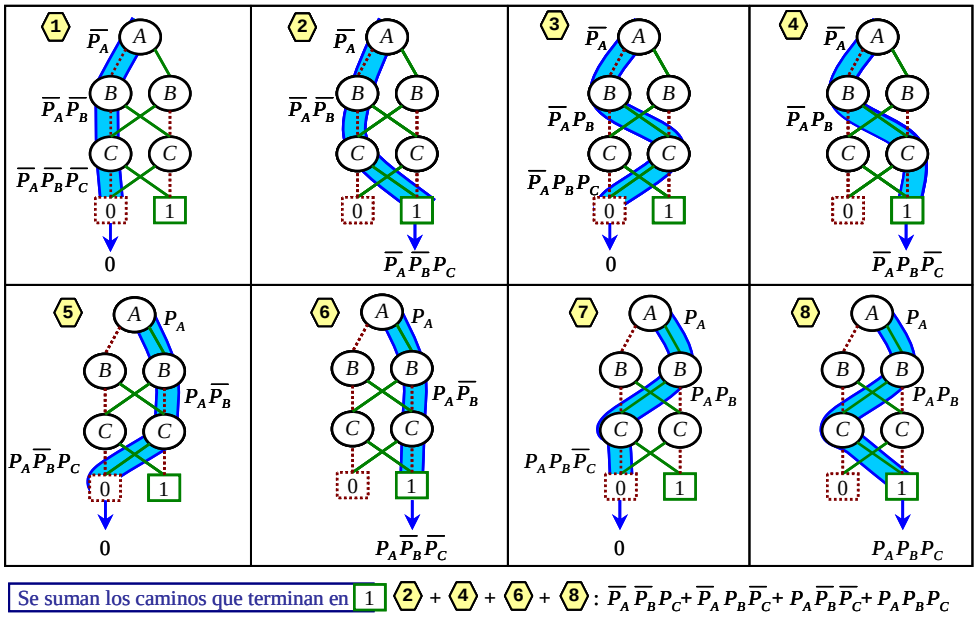


Figura AI-1: Cálculo descendente de la probabilidad de una XOR de tres entradas

La otra alternativa es ir anotando la probabilidad de los nodos del BDD conforme se recorren, evitando así continuar recorriendo caminos que ya tienen calculada su probabilidad. Este método se muestra en la figura AI-2 para el mismo ejemplo de la puerta XOR de tres entradas.

En la figura se ve que el primer camino se tiene que recorrer entero. En el paso segundo, una vez que el nodo *C* se ha recorrido por sus dos ramas (la rama cero no se ha contabilizado por terminar en el nodo final cero) se anota su probabilidad. Los caminos tercero y cuarto también se recorren enteros, tras recorrerlos ya se dispone de la probabilidad de otros dos nodos del BDD, éstos se han señalado en el paso cuarto. El quinto recorrido no hace falta terminarlo, ya que el nodo de la señal *C* ya tiene su probabilidad calculada, por tanto los caminos 5 y 6 de la figura AI-1 se convierten en un sólo camino en la figura AI-2. Igual ocurre con el sexto camino, pues la probabilidad del otro nodo de la señal *C* está anotada, reduciendo los caminos 7 y 8 del método descendente en un sólo camino (6). Al terminar los recorridos, todos los nodos tienen su probabilidad anotada y si este BDD perteneciese a un BDD mayor, al llegar a cualquiera de sus nodos no habría que seguir con el cálculo. Debido a que la probabilidad del BDD se va anotando en el recorrido de vuelta (subiendo) del BDD se ha denominado a este método cálculo ascendente.

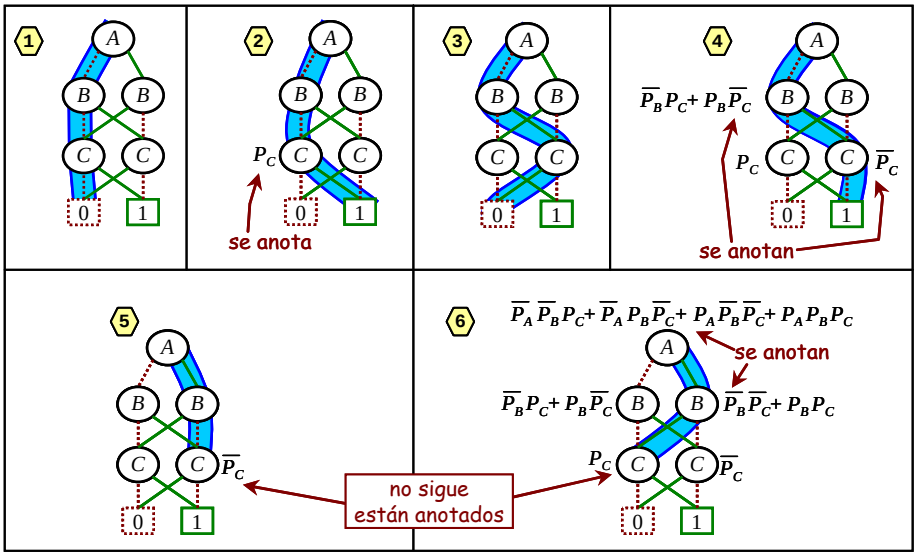


Figura AI-2: Cálculo ascendente de la probabilidad de una XOR de tres entradas

Es evidente que para este circuito sencillo el ahorro no es muy significativo, pero para circuitos grandes el ahorro de tiempo puede llegar a un orden de magnitud.

ANEXO II. ANEXOS A LA PROPUESTA

En este anexo se incluyen explicaciones más detalladas referentes al capítulo 3, que para facilitar la lectura del documento se han apartado.

AII.1 Algoritmos para la partición del circuito

En esta sección se desarrollan los algoritmos empleados en la partición del modelo del hardware. La explicación de concerniente a los fundamentos teóricos de la partición se ha realizado en el apartado 3.4.

El uso de las distintas capas del modelo simplificado y extendido del hardware (SHM y EHM) facilita el análisis de partición. Sobre estos modelos se identifica el área que se debe cubrir para cada señal.

El procedimiento para la partición del circuito es diferente según se vean involucradas sentencias condicionales o no, ya que si las hay se deben de tener en cuenta las regiones disjuntas que surgen a partir de ellas. Por tanto, a continuación se explicarán los procedimientos según sean sentencias condicionales o no.

AII.1.1 Partición en sentencias no condicionales

En este análisis se debe comprobar si cada una de las fuentes inmediatas de la señal analizada tienen dependencias comunes, en caso afirmativo se debe ir buscando hacia atrás hasta encontrar las señales que originan la reconvergencia.

El proceso se puede esquematizar en los siguientes pasos:

1. Buscar las fuentes inmediatas de la señal
2. Comprobar si algunas de las fuentes inmediatas tienen fuentes primarias comunes. Aquellas fuentes inmediatas que tengan una misma fuente primaria se añaden a una lista. Habrá una lista por cada fuente primaria
3. Se toma una de las listas y de ella la fuente inmediata con mayor profundidad combinacional, se saca de la lista, y se obtienen sus fuentes inmediatas. De estas nuevas fuentes inmediatas se añaden a la lista aquellas que tengan la misma dependencia primaria
4. Se vuelve a repetir el paso 3 hasta que la lista tenga la misma señal, que será la señal que origina la reconvergencia.

Con el fin de aclarar el procedimiento, se va a explicar mediante un ejemplo. En la figura AII-1 aparece un circuito de ejemplo en el que se han marcado las profundidades combinacionales de cada señal.

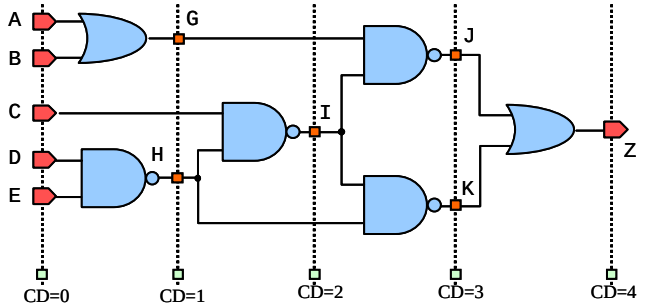


Figura AII-1: Circuito de ejemplo para el análisis de reconvergencia

En el primer paso hay que buscar las fuentes inmediatas de la señal Z, que es la señal que se va a estudiar. Para ello se toma la capa combinacional del SHM (§3.3.1) que ya las tiene apuntadas. De cada dependencia inmediata se necesita conocer sus fuentes primarias, que son obtenidas recurriendo a la capa secuencial. Por tanto, a partir de este primer análisis se obtendría el esquema mostrado en la figura AII-2, en el que ya se han identificado las fuentes primarias comunes (*C*, *D*, *E*) a las dependencias inmediatas (*J*, *K*).

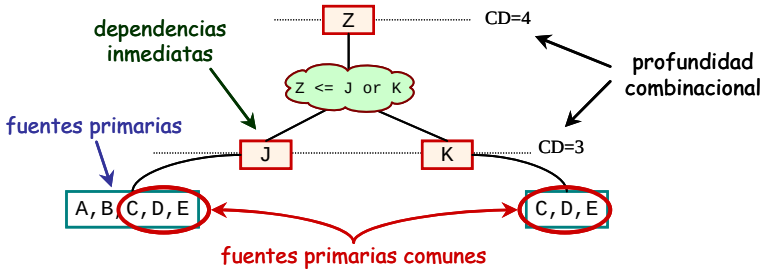


Figura AII-2: Obtención de las dependencias inmediatas de la señal Z

En el segundo paso, una vez que se han identificado dependencias inmediatas con fuentes primarias comunes, se agrupan en listas según la fuente primaria común de la que dependan. Para este caso saldrían tres listas correspondientes a las tres fuentes primarias comunes (*C*, *D*, *E*), y las tres con las mismas dependencias inmediatas (*J*, *K*). Estas listas se han representado en la figura AII-3.

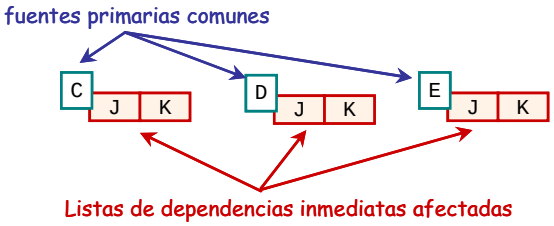


Figura AII-3: Obtención de las listas de dependencias inmediatas con fuente primaria común

En el tercer paso, se toma una lista y de ella la dependencia inmediata de mayor profundidad combinacional. Por ejemplo, se toma la lista de la fuente primaria *C*, y como *J* y *K* tienen la misma profundidad combinacional, se toma las dependencias inmediatas de una cualquiera de ellas, por ejemplo *J*. El resultado de este paso y la obtención de las listas de dependencias inmediatas se muestran en la figura AII-4. En esta tesis, la sustitución de una variable por sus fuentes inmediatas se llamará **composición de una señal**.

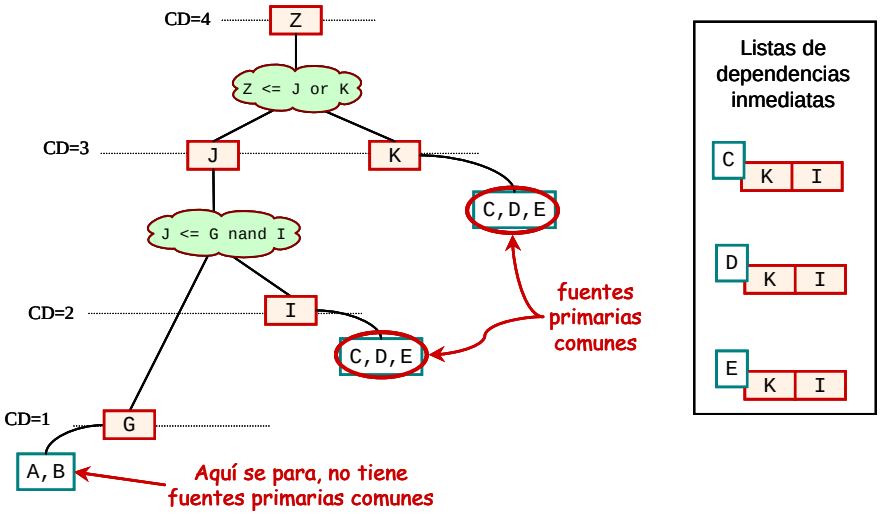


Figura AII-4: Obtención de dependencias inmediatas de J y listas de dependencias

Ahora se toma de nuevo la lista de dependencias con fuente final *C* y se obtienen las dependencias inmediatas de la señal *K* por tener mayor profundidad combinacional que *I*. La figura AII-5 muestra el resultado de esta operación.

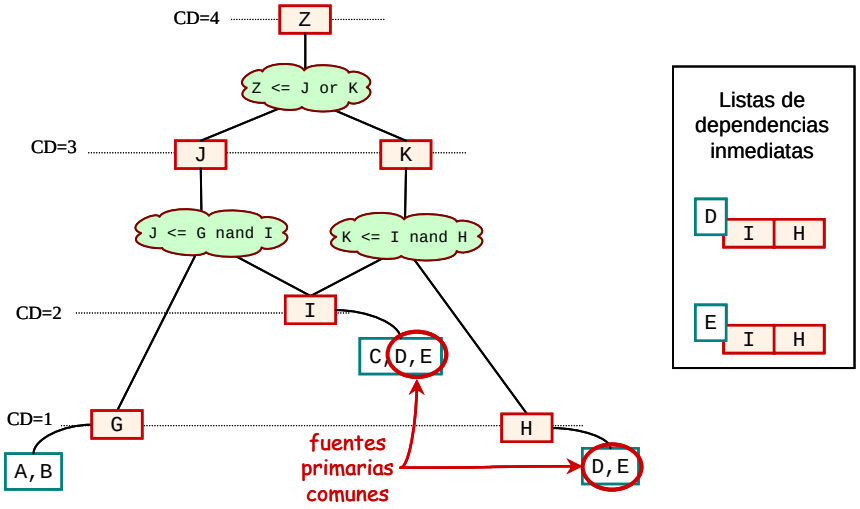


Figura AII-5: Obtención de las dependencias inmediatas de K

Al llegar a este punto, la fuente primaria *C* ha dejado de ser una fuente primaria común, y por tanto ya no tiene lista. Quedando solamente las listas de las fuentes primarias *D* y *E* (ver figura AII-5).

Por último, se buscan las dependencias inmediatas de la señal *I*, haciendo que termine el análisis, ya que ninguna las señales (*G*, *C*, *H*) tienen dependencias finales comunes (ver figura AII-6).

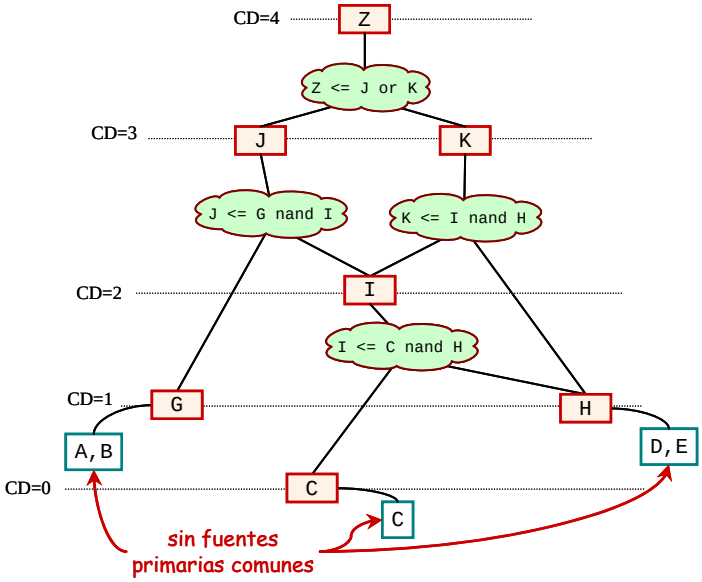


Figura AII-6: Fin del análisis, no hay fuentes primarias comunes

La región de reconvergencia resultante en el circuito representado en el nivel de puertas se ha dibujado en la figura AII-7.

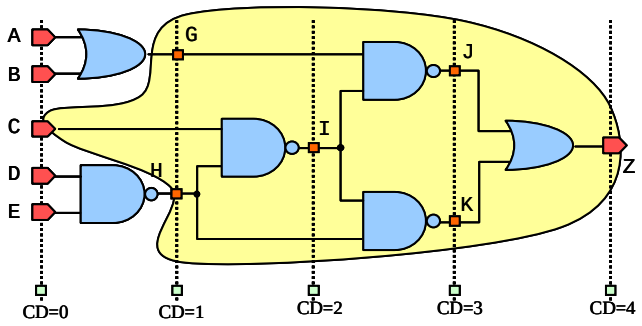


Figura AII-7: Región de reconvergencia resultante

Como se ha mostrado, el análisis no es complejo, sobre todo una vez que se dispone de las dependencias inmediatas y finales (o primarias) por medio de las capas secuencial y combinacional del SHM.

AII.1.2 Partición en sentencias condicionales

Ya se ha visto que las sentencias condicionales permiten, en muchos casos, la partición del circuito en regiones disjuntas que resultan menores que las regiones de reconvergencia. Por tanto, siempre que se cumplan las condiciones para que el error cometido sea aceptable (§3.4.3) se podrán realizar particiones disjuntas.

El análisis llevado a cabo para realizar esta partición utiliza el modelo extendido del hardware (EHM), ya que proporciona una estructura más adecuada para el análisis de los circuitos. El análisis es algo más complejo que el realizado en el apartado anterior debido a que el análisis de reconvergencia se debe de llevar a cabo dentro de una jerarquía de sentencias marcada por las condiciones.

La explicación del procedimiento se hará sobre el ejemplo de la figura 3-9. De la señal sobre la que se quiere realizar el análisis de reconvergencia (*Z*), se toman sus dependencias inmediatas. Para este caso, se trata de la condición más externa y de cada una de sus alternativas tomadas como un conjunto. Esto es, independientemente de su complejidad, cada alternativa se agrupa como si fuese una sentencia de asignación. De la misma manera, la expresión de la condición se considera como una expresión simple.

La figura AII-8 explica de manera gráfica esta abstracción para ejemplo tratado. La alternativa que se asigna a la señal *Z* cuando se cumple la condición es la sentencia *if* interna, y toda ella se ha sustituido por la señal *Alt1*. La alternativa que se asigna cuando la condición no se cumple se considera otra señal *Alt0*. Asimismo, la condición también se sustituye por la señal *Cond*. Esta condición se puede considerar una expresión, o bien, como se muestra en la figura, la señal *Cond* estaría definida como tipo *boolean*, y por tanto, se le puede asignar directamente la expresión de la condición y se puede colocar en la condición de la sentencia *if*.

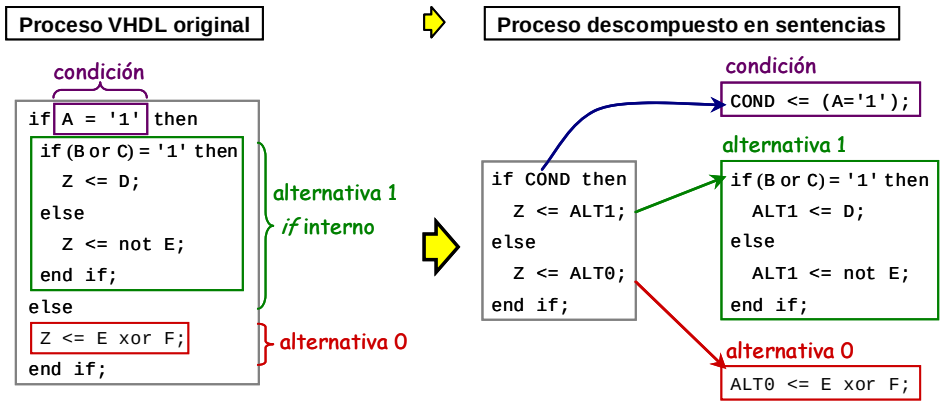


Figura AII-8: Descomposición en sus sentencias inmediatas de la asignación de una señal (Z) dentro de un proceso VHDL

Esta descomposición en las sentencias inmediatas de una señal asignada en una sentencia condicional quedaría representada en el EHM como se muestra en la figura AII-9. En ella, a partir de la señal *Z* se llega a la sentencia *if*, representada por un rombo. De este rombo sale a su derecha la condición *Cond* (*A*='1'); por debajo a la izquierda en línea discontinua, la alternativa resultante de la condición negada *Alt0* (*E* XOR *F*); y por abajo a la derecha, la alternativa resultante del cumplimiento de la condición. Esta última es a su vez, otra sentencia *if*, señalada en la figura como *if* interno, que se ha sustituido por la señal *Alt1*.

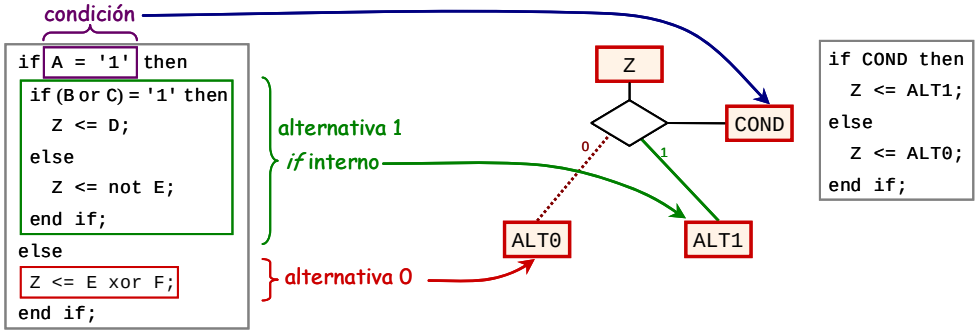


Figura AII-9: Descomposición de la asignación de una señal (Z) dentro de un proceso VHDL mostrada en el EHM

A la derecha de la figura AII-9 se muestra el proceso *virtual* al que directamente correspondería la representación del EHM mostrada.

Una vez que se tiene la representación del EHM de las sentencias inmediatas se realiza el análisis de reconvergencia. Para ello se analizan las fuentes primarias de cada una de las dependencias inmediatas. Estas fuentes primarias se ven en la figura AII-10. La condición no tiene ninguna fuente primaria común con las alternativas. Por tanto, se cumple la condición fundamental para realizar la partición (punto 1, §3.4.3). Así pues, a pesar de que las alternativas tienen una fuente común, como cuelgan de caminos disjuntos, no hace falta buscar el origen de la reconvergencia. Por otro lado, al haber sólo una fuente primaria común (*E*), de un total de cinco fuentes primarias (*B*, *C*, *D*, *E*, *F*) es de esperar que la simplificación tenga poca repercusión en el cálculo de la actividad (punto 4, §3.4.3).

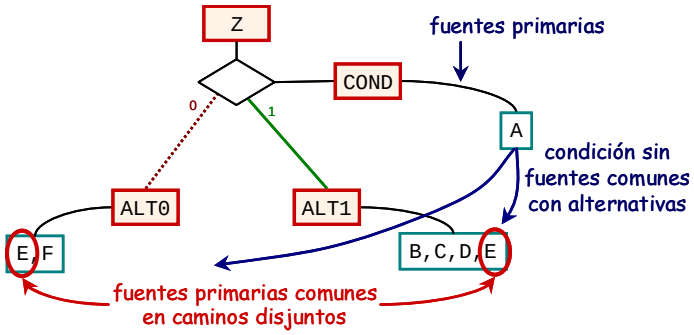


Figura AII-10: Primer paso del análisis de reconvergencia teniendo en cuenta las regiones disjuntas

Como resultado del análisis se obtiene una región de reconvergencia formada únicamente por tres señales (*Cond*, *Alt0*, *Alt1*). Sin embargo, debido a que la condición *Cond* sólo depende de la señal *A*, no se aumenta el número de dependencias si se sustituye *Cond* por su expresión original, quedando el EHM de la señal *Z* como se ve en la figura AII-11. La región disjunta resultante y su comparación con la región de reconvergencia ya se mostraron en la figura 3-20. En aquella figura, las señales *L* y *K* son lo que aquí se han llamado *Alt0* y *Alt1* respectivamente.

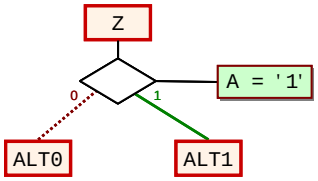


Figura AII-11: EHM resultante de la señal Z

Como las señales *Alt0* y *Alt1* han sido creadas para realizar el análisis, no existen en el circuito, y consecuentemente, no han sido analizadas. El análisis de la señal *Alt0* no tiene ninguna dificultad, ya que es una asignación en la que sus dependencias inmediatas son independientes. El análisis de la señal *Alt1* tampoco tiene mucha complejidad, ya que también todas sus dependencias inmediatas son independientes. Debido a que ambas alternativas dependen sólo de una señal, se ponen directamente, sin sustituirlas por una señal virtual. Por otro lado, al contrario de lo que ocurre con la condición del *if* exterior, la condición del *if* interno está formada por una expresión que involucra a dos señales. En consecuencia, para dejar la región lo más simplificada posible, se pone en función de una señal que es el resultado de dicha expresión (*Cond2*). Para clarificar, su esquema extendido (EHM) se ha dibujado en la figura AII-12.

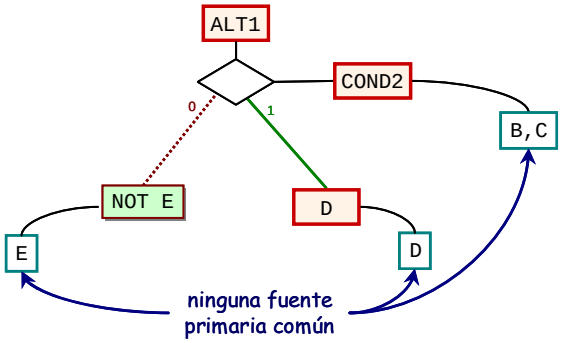


Figura AII-12: EHM resultante de la señal ALT1

Para concluir, en la figura AII-13 se ha dibujado todo el esquema extendido (EHM) del circuito marcando las particiones que se han llevado a cabo usando regiones disjuntas.

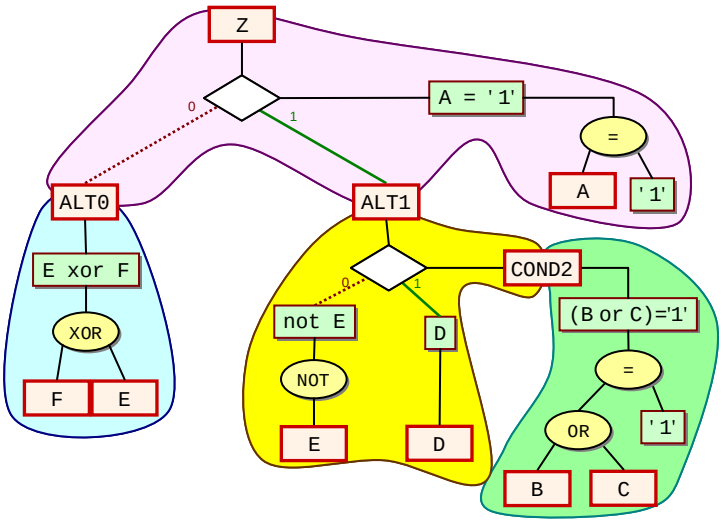


Figura AII-13: Partición del EHM mediante regiones disjuntas

Y la figura AII-14 muestra la partición del EHM llevada a cabo mediante regiones reconvergentes. Nótese cómo la señal *E* es la causante de la reconvergencia, y por tanto, tres de las regiones disjuntas de la figura AII-13 se han convertido en una sola región. En esta figura no aparecen las señales *Alt0*, *Alt1*, y *Cond2*, ya que no son necesarias para esta partición. Aún así, pueden ser accedidas internamente desde el EHM.

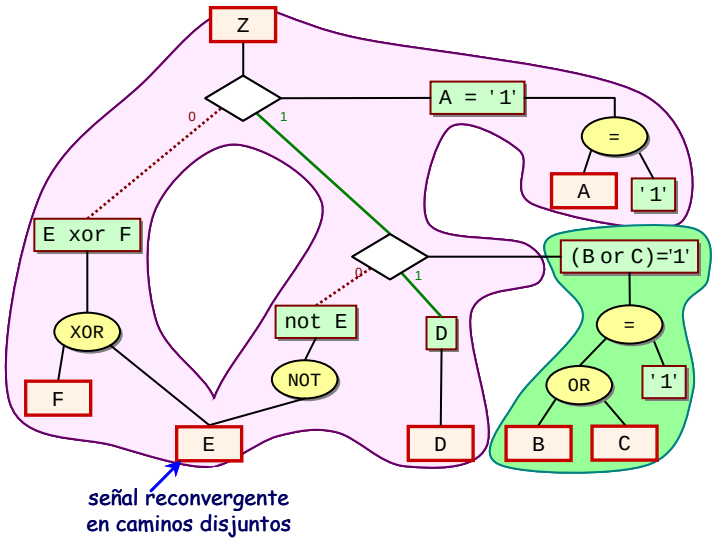


Figura AII-14: Partición del EHM mediante regiones reconvergentes

A continuación se pondrá otro ejemplo para clarificar el procedimiento en más situaciones.

Se tiene el proceso VHDL mostrado en la figura AII-15 y de él se extrae el EHM para sus dependencias inmediatas. Igual que el ejemplo anterior, estas dependencias inmediatas no son señales que se hayan declarado en el código, sino que se usan internamente en el EHM para realizar el análisis. En la figura se ha señalado la correspondencia de estas señales virtuales (*Alt1*, *Alt0*, *Cond1*) con el código.

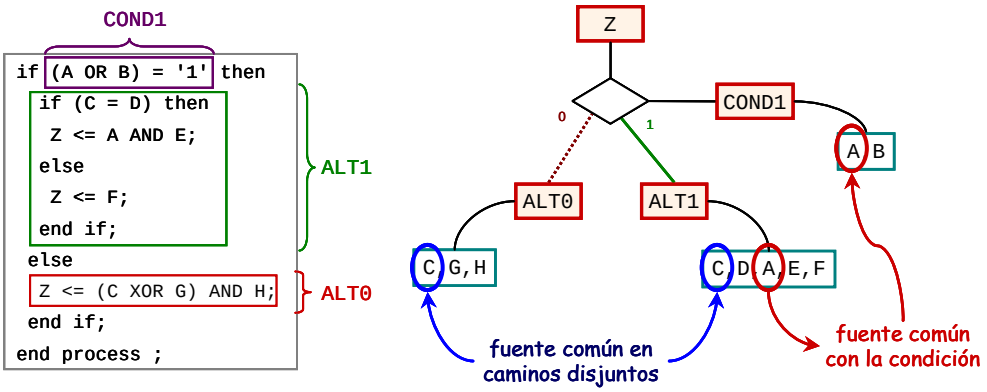


Figura AII-15: Primer paso del análisis de regiones disjuntas

En el EHM de la figura AII-15 se indican las fuentes primarias de cada dependencia inmediata. La señal *Alt0* tiene una fuente común (*C*) con la otra alternativa (*Alt1*), aún así, por estar en caminos disjuntos no es necesario componerla en sus fuentes inmediatas para buscar la fuente común. Sin embargo, para la alternativa *Alt1* no ocurre lo mismo debido a que tiene una fuente común con la condición *Cond1*. Por tanto, *Alt1* y *Cond1* deben ser compuestas en sus dependencias inmediatas.

El resultado de esta composición de señales aparece en la figura AII-16. En ella se señala la creación de otras tres señales virtuales: la condición de la sentencia *if* interna (*Cond2*) y sus dos alternativas[57] (*Salt0*, *Salt1*).

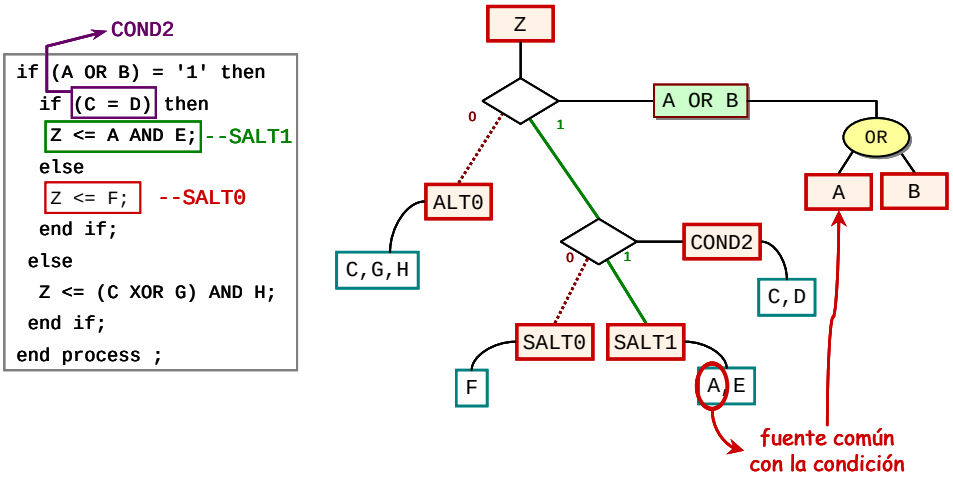


Figura AII-16: Resultado tras la composición de Cond1 y Alt1

La condición tiene la fuente *C* común con la señal *Alt0*, pero no se compone por estar en caminos disjuntos. La señal *Salt1* tiene la fuente común con la primera condición. Debido a que están en el mismo camino, se tendrá que componer. La señal *Salt0* no tiene ninguna fuente común, pero por depender de una sola señal (*F*) se sustituye por ella. El resultado final se ha dibujado en la figura AII-17.

[57] El nombre *Salt1* viene de *Sub-ALTernativa 1*; *sub* hace referencia a que es la alternativa de la sentencia *if* inferior (o interior). El uno o cero indica que es la alternativa que se asigna cuando se cumple (1) ó cuando no se cumple (0) la condición.

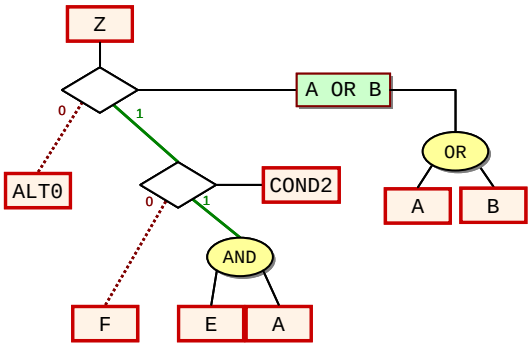


Figura AII-17: Área disjunta resultante en el EHM

Sin usar regiones disjuntas, el área de reconvergencia cubriría todo el diseño, quedando como se puede apreciar en la figura AII-18. Comparando las figuras se pueden evaluar las diferencias. En el capítulo de resultados experimentales (capítulo 4) se contabiliza la repercusión en el coste computacional y de memoria.

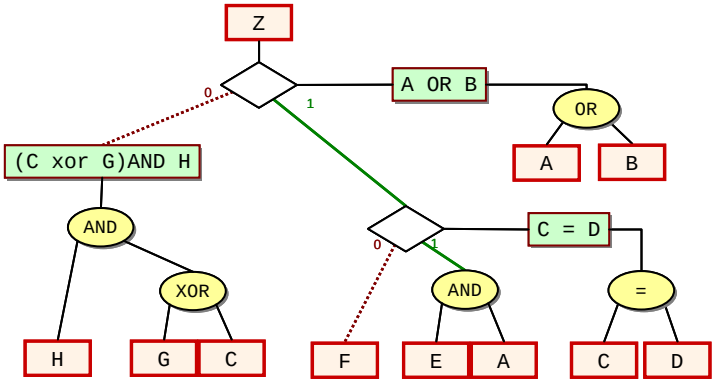


Figura AII-18: Área de reconvergencia resultante en el EHM

AII.2 Ejemplos del error en el cálculo de actividad con regiones disjuntas

Este apartado del anexo se incluye con el fin de facilitar el entendimiento acerca de los errores en el cálculo de la actividad con regiones disjuntas. El contenido de este anexo complementa al apartado 3.4.3. A continuación se mostrará el análisis del error cometido para algunos ejemplos. Estos ejemplos tienen un multiplexor cuyas alternativas tienen dependencias comunes.

AII.2.1 Ejemplo primero

Para el primer ejemplo se recurrirá a un caso extremo de dependencia en el que ambas alternativas son en realidad la misma señal (figura AII-19). Este ejemplo es meramente teórico, ya que el análisis automático que se realiza al hacer la partición detecta[58] esta situación y asigna directamente la señal *B* a la salida *Z*. Este ejemplo se pone con el fin de conocer el error para el peor caso de dependencia entre las alternativas.

[58] Además, la herramienta advierte al diseñador de que la sentencia es conflictiva ya que puede conducir a errores o que quizá haya habido un error en la codificación del diseño VHDL

```
I <= B;
J <= B;
P:process(A,I,J)
begin
 if A = '1' then
  Z <= I;
 else
  Z <= J;
 end if;
end process;
```

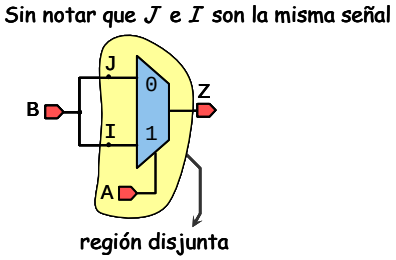


Figura AII-19: Ejemplo primero: Ejemplo extremo de dependencias entre las alternativas del multiplexor

Evidentemente la actividad de *Z* va a ser la misma que la actividad de *B*. Por tanto, la señal *Z* no se verá afectada por la señal *A*. Sin embargo se va a suponer que se crea la región disjunta como se muestra en la figura AII-19, y que se desconoce que *J* e *I* se corresponden con la misma señal.

Por tanto, se calculan los cofactores para comparar los resultados proporcionados por el cálculo mediante las regiones disjuntas con los cálculos exactos.

Los cofactores relacionados con la inactividad de *A* no tienen error y tanto para el método exacto como el aproximado, son:

$$a(Z)_{A_{0\to0}} = a'(Z)_{A_{0\to0}} = a_J = a_B$$
$$a(Z)_{A_{1\to1}} = a'(Z)_{A_{1\to1}} = a_I = a_B$$

La figura AII-20 compara el cálculo del cofactor $a(Z)_{A_{0\to1}}$ por el método aproximado y por el método exacto. Nótese que por ser un caso extremo, el método exacto es más sencillo. En la mayoría de los casos, la sustitución por las fuentes independientes es más trabajosa y para circuitos grandes, los requerimientos computacionales y de memoria pueden ser prohibitivos.

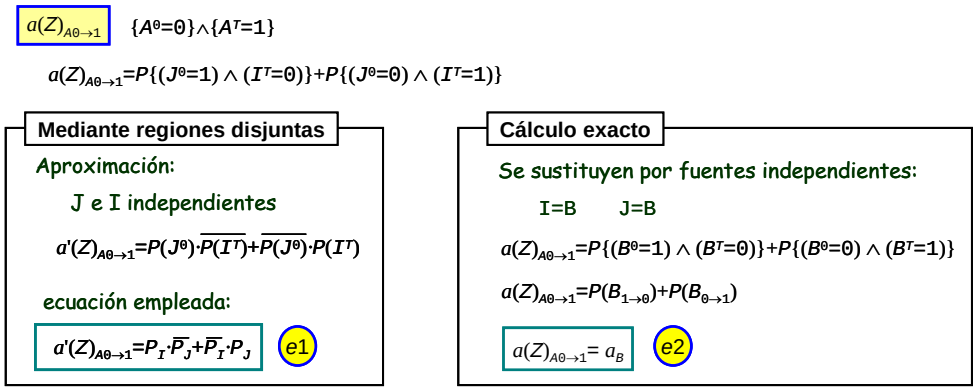


Figura AII-20: Cálculo de un cofactor de actividad para el ejemplo de la figura AII-19

El proceso de cálculo para el otro cofactor $a(Z)_{A_{1\to0}}$ es idéntico y da el mismo resultado, quedando la ecuación del método aproximado (*e*1 de la figura AII-20) para ambos:

$$a'(Z)_{A_{0\to1}} = a'(Z)_{A_{1\to0}} = P_I \cdot \overline{P}_J + \overline{P}_I \cdot P_J \qquad \text{(AII.1)}$$

Mientras que por el método el exacto quedan (*e*2 de la figura AII-20):

$$a(Z)_{A_{0\to1}} = a\,(Z)_{A_{1\to0}} = a_B \qquad \text{(AII.2)}$$

Con el fin de comparar la ecuación del cofactor de actividad aproximado (ecuación AII.1) con la del exacto (ecuación AII.2), se pondrá la ecuación AII.1 en función de la señal *B*. Como $P_I = P_J = P_B$, queda:

$$a'(Z)_{A_{0\to1}} = a'(Z)_{A_{1\to0}} = P_B \cdot \overline{P}_B + \overline{P}_B \cdot P_B = 2 \cdot P_B \cdot \overline{P}_B \qquad \text{(AII.3)}$$

Observando las ecuaciones de los cofactores de actividad aproximado y exacto (AII.3 y AII.2) se puede apreciar que para su cálculo aproximado (AII.3) no se tiene en cuenta la dependencia temporal de *B* (ver ecuación 2.21) y por tanto, al considerarla temporalmente independiente se separa en la probabilidad de la señal multiplicada por su probabilidad negada.

Sustituyendo los cofactores en la ecuación 3.2, la ecuación aproximada de la actividad de la salida *Z* del multiplexor queda:

$$a'_Z = P(A_{1\to1})\cdot a_B + P(A_{0\to0})\cdot a_B + a_A \cdot 2 \cdot P_B \cdot \overline{P_B}$$

Que por la ecuación 2.6 resulta:

$$a'_Z = \overline{a}_A \cdot a_B + a_A \cdot (2 \cdot P_B \cdot \overline{P_B}) \quad \text{(AII.4)}$$

Aunque la actividad exacta de *Z* es igual a la actividad de *B*, con el fin de compararla con la ecuación AII.4, se puede poner en la forma equivalente:

$$a_Z = a_B = \overline{a}_A \cdot a_B + a_A \cdot a_B \quad \text{(AII.5)}$$

Donde, por la comparación de las ecuaciones AII.4 y AII.5 se puede apreciar la aproximación realizada. En este caso, el error cometido está causado por la dependencia temporal que tenga la fuente común (*B*) y su influencia disminuirá cuanto menor sea la actividad de la señal de selección (*A*).

El peor caso resultará cuando la señal de selección tenga máxima actividad $a_A = 1$, en estas condiciones el error que se cometerá vendrá dado exclusivamente por el error cometido por no considerar las correlaciones temporales de la señal *B*.

AII.2.2 Ejemplo segundo

En el segundo ejemplo que se pondrá, hay una fuerte dependencia entre ambas alternativas, aunque no tan acusada como en el primer ejemplo. El diseño se muestra en la figura AII-21.

```
if A = '1' then
  Z <= B and C;
else
  Z <= C;
end if;
```

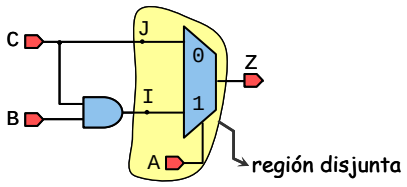


Figura AII-21: Ejemplo segundo: Fuerte dependencia de las alternativas en una señal común

Los cofactores de inactividad siempre dan el mismo resultado numérico con el cálculo mediante regiones disjuntas y el exacto:

$$a(Z)_{A_{0\to0}} = a'(Z)_{A_{0\to0}} = a_J = a_C \quad \text{(AII.6)}$$

$$a'(Z)_{A_{1\to1}} = a_I \quad \text{(AII.7)}$$

$$a(Z)_{A_{1\to1}} = P(B_{0\to1})\,P(C^T) + P(B_{1\to1})\,P(C_{0\to1}) + P(B_{1\to1})\,P(C_{1\to0}) + P(B_{1\to0})\,P(C^0) \quad \text{(AII.8)}$$

Bajo la asunción de proceso estrictamente estacionario (ecuación 2.2), queda:

$$a(Z)_{A_{1\to1}} = P(B_{0\to1})\,P_C + P(B_{1\to1})\,P(C_{0\to1}) + P(B_{1\to1})\,P(C_{1\to0}) + P(B_{1\to0})\,P_C \quad \text{(AII.9)}$$

La fórmula AII.9 corresponde con la ecuación de la actividad de una puerta AND. Se ha incluido para compararla con la ecuación AII.7, y así mostrar el ahorro que proporciona el cálculo de la actividad mediante regiones disjuntas.

Para la obtención de la ecuación de los cofactores de actividad se parte de la ecuación:

$$a(Z)_{A_{0\to1}} = P\{J^0 \wedge \neg I^T\} + P\{\neg J^0 \wedge I^T\} \quad \text{(AII.10)}$$

Si se sigue el método aproximado se realiza igual que en el ejemplo anterior, sin considerar las dependencias temporales, quedando por tanto:

$$a'(Z)_{A_{0\to1}} = a'(Z)_{A_{1\to0}} = P_I \cdot \overline{P_J} + \overline{P_I} \cdot P_J \quad \text{(AII.11)}$$

Mientras que calcularla por el método exacto tienen más complicación, ya que se tiene que sustituir *J* e *I* por sus fuentes independientes:

$$a(Z)_{A_{0\to1}} = P\{C^0 \wedge \neg(B^T \wedge C^T)\} + P\{\neg C^0 \wedge (B^T \wedge C^T)\}$$

Aplicando la ley de De Morgan, $\{\neg(B^T \wedge C^T) = \neg B^T \vee \neg C^T\}$:

$$a(Z)_{A_{0\to1}} = P\{C^0 \wedge (\neg B^T \vee \neg C^T)\} + P\{\neg C^0 \wedge C^T \wedge B^T\}$$

Y como

$$(\neg B^T \vee \neg C^T) = \{\neg B^T \vee (B^T \wedge \neg C^T)\}$$

queda:

$$a(Z)_{A_{0\to1}} = P\{C^0 \wedge (\neg B^T \vee (B^T \wedge \neg C^T))\} + P\{\neg C^0 \wedge C^T \wedge B^T\}$$

Aplicando la propiedad distributiva:

$$a(Z)_{A_{0\to1}} = P\{(C^0 \wedge \neg B^T) \vee (C^0 \wedge \neg C^T \wedge B^T))\} + P\{\neg C^0 \wedge C^T \wedge B^T\}$$

El primer sumando se puede separar debido a que es un OR de dos términos disjuntos:

$$a(Z)_{A_{0\to1}} = P\{C^0 \wedge \neg B^T\} + P\{C^0 \wedge \neg C^T \wedge B^T\} + P\{\neg C^0 \wedge C^T \wedge B^T\}$$

Como se supone que las entradas (*B*, *C*) son independientes, y son procesos estacionarios:

$$a(Z)_{A_{0\to1}} = P_C \cdot \overline{P_B} + P(C_{1\to0}) \cdot P_B + P(C_{0\to1}) \cdot P_B$$

Y mediante la ecuación 2.5, finalmente ambos cofactores de actividad quedan:

$$a(Z)_{A_{0\to1}} = a(Z)_{A_{1\to0}} = \overline{P_B} \cdot P_C + P_B \cdot a_C \qquad \text{(AII.12)}$$

La ecuación AII.12 correspondiente los cofactores de actividad calculados por el método exacto tiene una complejidad similar a la calculada por el método aproximado (AII.11) ya que ambas están en función del mismo número de señales. Como se verá más adelante, esto se debe a que el número de fuentes independientes de las alternativas es pequeño y número de fuentes comunes es grande respecto a las fuentes independientes.

Sin embargo, se ha podido apreciar que mientras la ecuación del método aproximado es directa, la del método exacto requiere un procesamiento matemático, que cuya elaboración se complica con el tamaño del circuito.

Nuevamente, para comparar el error cometido en el método aproximado, se sustituirá la ecuación AII.11 por los valores de probabilidad referidos a las variables independientes *B* y *C*, que son:

$$P_I = P_B \cdot P_C$$
$$P_J = P_C$$
$$\overline{P_I} = \overline{P_B} + P_B \cdot \overline{P_C}$$
$$\overline{P_J} = \overline{P_C}$$

Y por tanto queda:

$$a'(Z)_{A_{0\to1}} = a'(Z)_{A_{1\to0}} = \overline{P_B} \cdot P_C \cdot + 2 \cdot P_B \cdot P_C \cdot \overline{P_C} \qquad \text{(AII.13)}$$

Una vez más se ve que la diferencia entre el cofactor de actividad calculado por el método exacto (ecuación AII.12) y el aproximado (AII.13) estriba en la no consideración de las dependencias temporales de la señal común (*C*).

Para comparar de manera visual ambos procedimientos, la figura AII-22 muestra de manera resumida la obtención de ambas ecuaciones y el error cometido en el método aproximado.

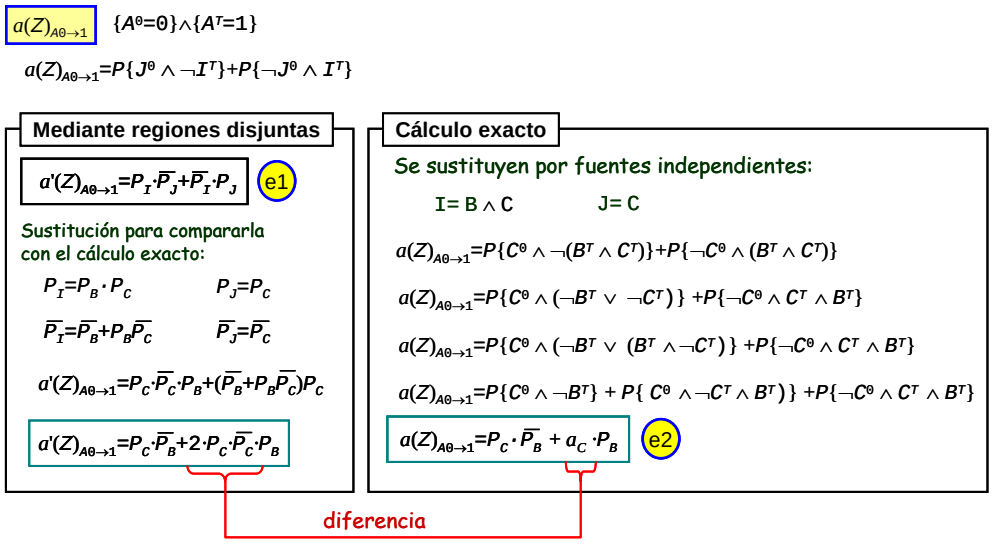


Figura AII-22: Resumen de los cálculos de los cofactores de actividad para el ejemplo de la figura AII-21

La sustitución de los cofactores calculados por el método de regiones disjuntas en la ecuación de la actividad (ecuación 3.2), da como resultado:

$$a'_Z = P(A_{1\to1})\cdot a_I + P(A_{0\to0})\cdot a_C + a_A(P_C\cdot\overline{P}_B + 2\cdot P_C\cdot\overline{P}_C\cdot P_B) \qquad \text{(AII.14)}$$

Y empleando el método exacto:

$$a_Z = P(A_{1\to1})\cdot a_I + P(A_{0\to0})\cdot a_C + a_A(P_C\cdot\overline{P}_B + a_C\cdot P_B) \qquad \text{(AII.15)}$$

De nuevo se puede comprobar que el error depende de la actividad de la señal de selección (*A*), y en este caso, el error tiene menor influencia en el último sumando, pues se encuentra multiplicado y junto con otros sumandos. Por tanto, cuanto menor sea la probabilidad de la señal *B*, menor influencia tendrá.

Se debe señalar que las ecuaciones AII.14 y AII.15 son ilustrativas, y se han ajustado para que sean fácilmente comparables entre sí, poniendo las ecuaciones en función de los mismos términos. Éstas no son las ecuaciones que se hubiesen empleado en el cálculo, ya que la ecuación obtenida a través de las regiones disjuntas habría estado en función de las señales *A*, *I* y *J*. Mientras que la ecuación del método exacto habría estado en función de *A*, *B* y *C*.

AII.2.3 Ejemplo tercero

Para el tercer ejemplo se pondrá un multiplexor en el que sus dos alternativas dependen de las mismas variables. El código y su representación en puertas están en la figura AII-23, donde también se ha señalado la región disjunta.

```
I <= B and C;
J <= B or C;
P:process(A,I,J)
begin
 if A = '1' then
  Z <= I;
 else
  Z <= J;
 end if;
end process;
```

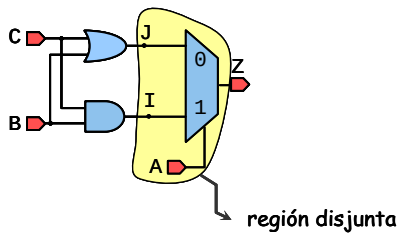


Figura AII-23: Ejemplo tercero: Las alternativas dependen de las mismas señales

El proceso seguido para la obtención de las ecuaciones es similar a los de los ejemplos anteriores, y por tanto para simplificar sólo se pondrán las ecuaciones resultantes.

$$a'_Z = P(A_{1\to1})\cdot a_I + P(A_{0\to0})\cdot a_J +$$

$$+ a_A(\overline{P_B}\cdot\overline{P_B}\cdot P_C + P_B\cdot P_B\cdot\overline{P_C} + 2\cdot P_B\cdot\overline{P_B}\cdot P_C\cdot\overline{P_C} + P_B\cdot\overline{P_B}) \quad \text{(AII.16)}$$

Y empleando el método exacto:

$$a_Z = P(A_{1\to1})\cdot a_I + P(A_{0\to0})\cdot a_J +$$
$$+ a_A[P(B_{0\to0})P_C + P(B_{1\to1})\cdot\overline{P_C} + \tfrac{1}{2}\cdot a_B\cdot a_C + \tfrac{1}{2}\cdot a_B] \quad \text{(AII.17)}$$

A partir de las ecuaciones aproximada (AII.16) y exacta (AII.17) se puede extraer el error cometido. Igual que en los anteriores casos, el error depende de la actividad de la señal de selección (a_A). A diferencia del ejemplo anterior, para este caso, todos los sumandos del cofactor de actividad (el sumando afectado por a_A) tienen un error debido a no considerar dependencias temporales. Puede concluirse que al aumentar el número de señales comunes el error aumenta.

AII.2.4 Ejemplo cuarto

El cuarto ejemplo, mostrado en la figura AII-24, tiene una estructura similar al anterior, sin embargo, las alternativas comparten una sola dependencia (*D*).

```
I <= B and D;
J <= C or D;
P:process(A,I,J)
begin
 if A = '1' then
  Z <= I;
 else
  Z <= J;
 end if;
end process;
```

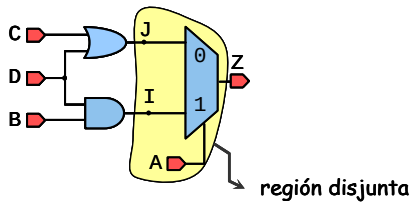


Figura AII-24: Ejemplo cuarto: Las alternativas comparten una sola dependencia

Las ecuaciones se obtienen de manera similar a los anteriores ejemplos. Para el método aproximado resulta:

$$a'_Z = P(A_{1\to1})\cdot a_I + P(A_{0\to0})\cdot a_J +$$
$$+ a_A(\overline{P_B}\cdot\overline{P_C}\cdot P_D + P_B\cdot P_C\cdot\overline{P_D} + P_C\cdot\overline{P_B} + P_B\cdot\overline{P_C}\cdot 2\cdot P_D\cdot\overline{P_D}) \quad \text{(AII.18)}$$

Y empleando el método exacto:

$$a_Z = P(A_{1\to1})\cdot a_I + P(A_{0\to0})\cdot a_J +$$
$$+ a_A(\overline{P_B}\cdot\overline{P_C}\cdot P_D + P_B\cdot P_C\cdot\overline{P_D} + P_C\cdot\overline{P_B} + P_B\cdot\overline{P_C}\cdot a_D) \quad \text{(AII.19)}$$

La diferencia entre las ecuaciones AII.18 y AII.19 es mucho menor que para el ejemplo anterior (ecuaciones AII.16 y AII.17). Sólo el último sumando del cofactor de actividad es aproximado. Por lo tanto, ahora el error disminuye con actividades bajas de la señal de selección, *A*, y con probabilidades bajas para la señal *B* y probabilidades altas para la señal *C*.

AII.2.5 Ejemplo quinto

Ahora se mostrarán las ecuaciones resultantes y los errores cometidos para el ejemplo vistos en la figura 3-19.

La ecuación obtenida por el método aproximado para el ejemplo de la figura 3-19 resulta:

$$a'_Z = P(A_{1\to1})\cdot a_J + P(A_{0\to0})\cdot a_K +$$
$$+ a_A[\overline{P_G}\cdot\overline{P_C}\cdot P_H + P_G\cdot P_C\cdot\overline{P_C} +$$
$$+ (P_G\cdot\overline{P_C}\cdot\overline{P_C}\cdot\overline{P_H} + P_G\cdot P_C\cdot P_C\cdot\overline{P_H}) +$$
$$+ (P_G\cdot P_C\cdot\overline{P_C}\cdot P_H\cdot P_H + P_G\cdot P_C\cdot\overline{P_C}\cdot\overline{P_H}\cdot\overline{P_H})] \quad \text{(AII.20)}$$

Y la ecuación obtenida mediante el método exacto queda:

$$a_Z = P(A_{1\to1})\cdot a_J + P(A_{0\to0})\cdot a_K +$$
$$+ a_A(\overline{P_G}\cdot\overline{P_C}\cdot P_H + ½\cdot P_G\cdot a_C + P_G\cdot\overline{a_C}\cdot\overline{P_H} + ½\cdot P_G\cdot a_C\cdot\overline{a_H}) \quad \text{(AII.21)}$$

En este ejemplo existe una dependencia muy fuerte entre las alternativas del multiplexor, ya que todas las dependencias de la alternativa *K* influyen en la alternativa *J*. Por tanto, tres de los cuatro sumandos del cofactor de actividad tienen términos aproximados. Para apreciar mejor la aproximación, en la figura AII-25 se muestran las diferencias entre los cofactores de actividad.

Como en todos los casos, el error disminuye cuanto menor es la actividad de la señal de selección a_A. Para este caso, además, el error disminuye cuanto menor es la probabilidad de la señal *G*.

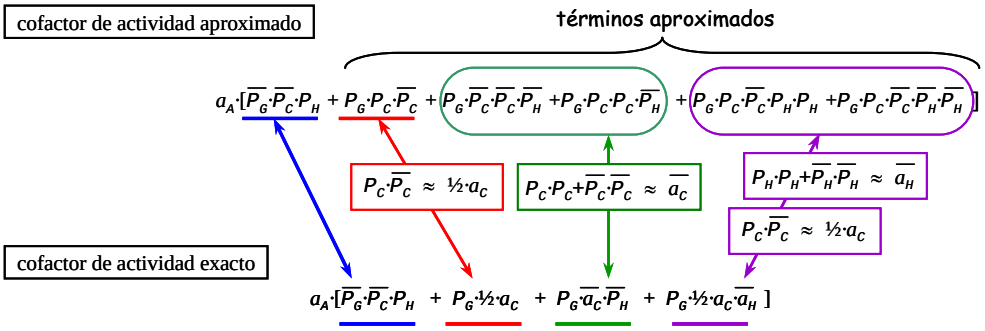


Figura AII-25: Comparación de los cofactores de actividad del ejemplo de la figura 3-19 calculados por el método aproximado y exacto (ecuaciones AII.20 y AII.21)

Recuérdese que, igual para los ejemplos anteriores, la ecuación AII.20 está desplegada para comparar los errores con la ecuación AII.21. Mientras que por el mismo motivo, la ecuación AII.21 está simplificada. Por tanto, no se deben comparar con criterios de complejidad, sino por los valores numéricos que resulten.

AII.2.6 Ejemplo sexto

El último ejemplo corresponde con el de la figura 3-20, cuya región de reconvergencia tiene 5 entradas y de las cuales, sólo la señal *E* es común a ambas alternativas. Por estas características, el error cometido es menos importante que en el ejemplo anterior. Y así, la actividad para el método aproximado resulta:

$$a'_Z = P(A_{1\to1})\cdot a_K + P(A_{0\to0})\cdot a_L +$$
$$+ a_A[\overline{P_M}\cdot P_F\cdot 2\cdot P_E\cdot\overline{P_E} + \overline{P_M}\cdot\overline{P_F}\cdot(P_E\cdot P_E+\overline{P_E}\cdot\overline{P_E}) +$$
$$+ P_M\cdot P_D\cdot P_F\cdot P_E + P_M\cdot P_D\cdot\overline{P_F}\cdot\overline{P_E} + P_M\cdot\overline{P_D}\cdot\overline{P_F}\cdot P_E + P_M\cdot\overline{P_D}\cdot P_F\cdot\overline{P_E}] \quad \text{(AII.22)}$$

Y empleando el método exacto:

$$a_Z = P(A_{1\to1})\cdot a_K + P(A_{0\to0})\cdot a_L +$$
$$+ a_A[\overline{P_M}\cdot P_F\cdot a_E + \overline{P_M}\cdot\overline{P_F}\cdot\overline{a_E} +$$
$$+ P_M\cdot P_D\cdot P_F\cdot P_E + P_M\cdot P_D\cdot\overline{P_F}\cdot\overline{P_E} + P_M\cdot\overline{P_D}\cdot\overline{P_F}\cdot P_E + P_M\cdot\overline{P_D}\cdot P_F\cdot\overline{P_E}] \quad \text{(AII.23)}$$

Comparando las ecuaciones AII.22 y AII.23 se puede apreciar que la simplificación tiene menor repercusión en el cálculo de la actividad. De los seis sumandos del cofactor de actividad sólo dos están influenciados por la simplificación. Por otro lado, el error disminuye conforme la probabilidad de *M* aumenta. Para facilitar la comparación, la figura AII-26 muestra en qué términos del cofactor de actividad se realiza la simplificación.

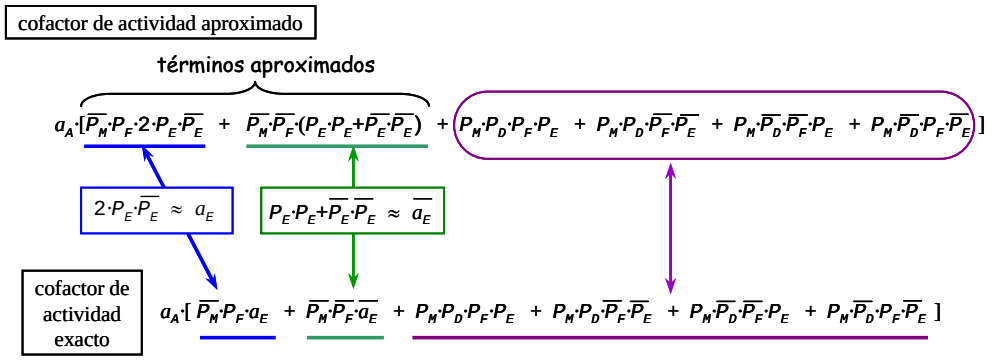


Figura AII-26: Comparación de los cofactores de actividad para el ejemplo de la figura 3-20 calculados por el método aproximado y exacto (ecuaciones AII.22 y AII.23)

AII.3 Construcción de los BDD de probabilidad

Este apartado del anexo explica cómo se construyen los diagramas de decisión binaria (BDD) de probabilidad a partir del EHM. Los BDD de probabilidad se utilizan para obtener la ecuación de la probabilidad de cada nodo del circuito. El resultado de esta ecuación se utilizará para el cálculo de la actividad.

Antes de construir los BDD de probabilidad se debe haber creado el EHM del circuito (§3.3), y haber realizado su partición (§3.4).

Los fundamentos y beneficios de los BDD en el cálculo de la probabilidad se han explicado la sección 2.2.2. Existen otros métodos diferentes a los BDD para este cálculo, como pueden ser la representación polinómica [103] y las redes bayesianas [6]. Sin embargo, el amplio uso de los BDD y la existencia de herramientas de código abierto para su manipulación han contribuido a que se hayan escogido los BDD.

El BDD empleado para el cálculo de la probabilidad de una función lógica es el mismo que el BDD de la propia función lógica (§2.2.2). La construcción de los BDD se realiza recorriendo el EHM, transformando las expresiones VHDL en BDD, y componiendo los BDD de las expresiones en los BDD ya construidos. La construcción y composición de los BDD se realiza automáticamente mediante la integración del paquete BuDDy [64] en la herramienta CAD (anexo IV) que se ha elaborado en la realización de esta tesis doctoral.

El paquete BuDDy es una biblioteca de código abierto y licencia de dominio público que proporciona la mayor parte de las funciones empleadas en la creación y manipulación de los BDD. El hecho de tener una licencia de domino público y el código fuente abierto permite acceder y modificar la biblioteca para adaptar y crear las operaciones específicas destinadas al cálculo de la probabilidad y actividad.

Para enseñar cómo se realiza el procedimiento en la práctica, se hará sobre el mismo ejemplo de la figura 3-9, cuyo EHM resultante está en la figura AII-17. Para facilitar la explicación se empleará el EHM representado en la figura AII-27, en el que se han incluido las señales virtuales intermedias. También se ha vuelto a incluir el código VHDL para tenerlo a la vista.

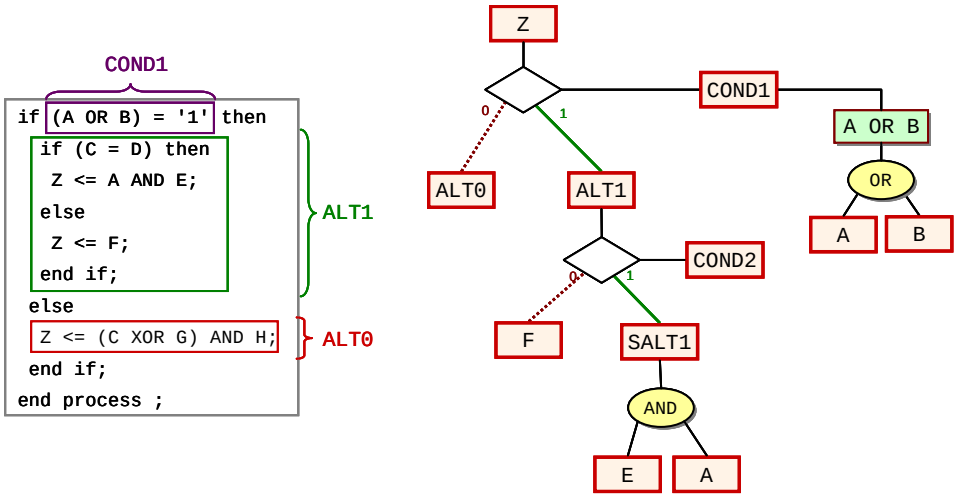


Figura AII-27: EHM de la figura AII-17 en el que se han incluido las señales virtuales intermedias

Para empezar a crear el BDD de probabilidad, se toma el nodo EHM de la señal que se quiere analizar (Z) y se examina la sentencia que de él cuelga. En este caso se trata de una sentencia condicional (*if-then-else*, para resumir: ITE), para la que existe una función en el paquete BuDDy. Así se crea la primera estructura del BDD, formada por las señales virtuales (*Cond1*, *Alt0*, *Alt1*), véase la figura AII-28.

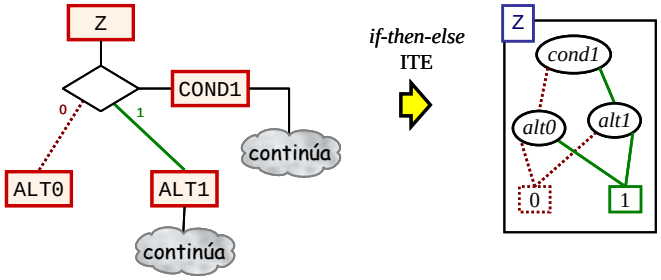


Figura AII-28: Primer paso para la creación del BDD del ejemplo de la figura AII-27

Según el EHM de la figura AII-27, tanto la condición *Cond1* como la alternativa *Alt1* se deben componer, y por eso en la figura AII-28 se han incluido unas nubecitas que indican que por ellas se debe continuar el proceso de creación del BDD.

Así, se continuaría con la creación del BDD de la condición *Cond1*, aplicando directamente la función de BuDDy que realiza esta operación (véase la figura AII-29).

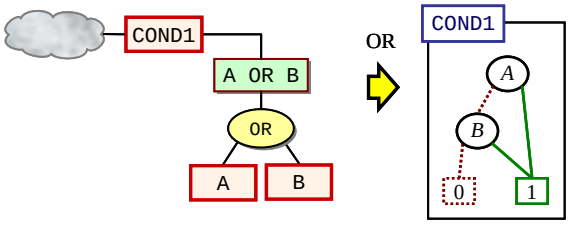


Figura AII-29: Creación del BDD de la operación de la señal Cond1

Y por último se crean los BDD que cuelgan de la señal *Alt1*. A la izquierda de la figura AII-30 está el BDD de la sentencia *if* interior; y a la derecha la creación del BDD de la operación AND, que corresponde con la señal *Salt1*.

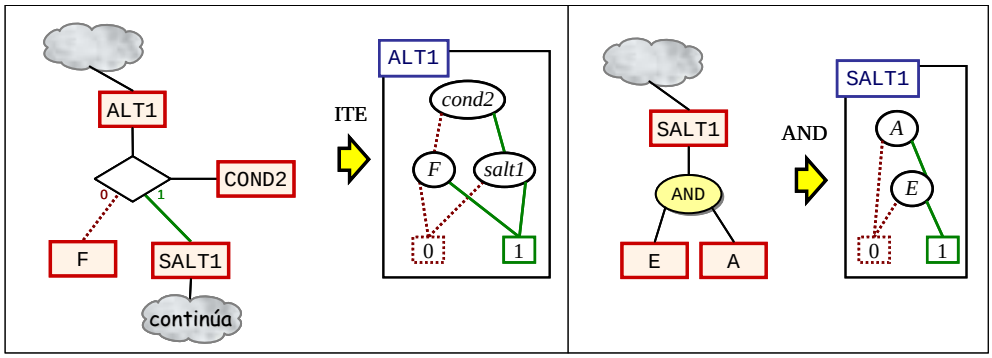


Figura AII-30: Creación de los BDD de las estructuras que cuelgan de la señal Alt1

Se acaba de mostrar la creación de los BDD individuales de cada una de las partes del EHM de la figura AII-27, sin embargo, en el proceso de construcción del BDD, a la vez que se van creando los BDD individuales se van componiendo unos con otros. La composición de BDD es una operación habitual para los BDD, en los que se sustituye una variable de un BDD por el BDD correspondiente a dicha variable.

Por ejemplo en la figura AII-31 se muestra la composición del BDD de la variable *Salt1* en el BDD de *Alt1*, que son los BDD que se han creado en la figura AII-30.

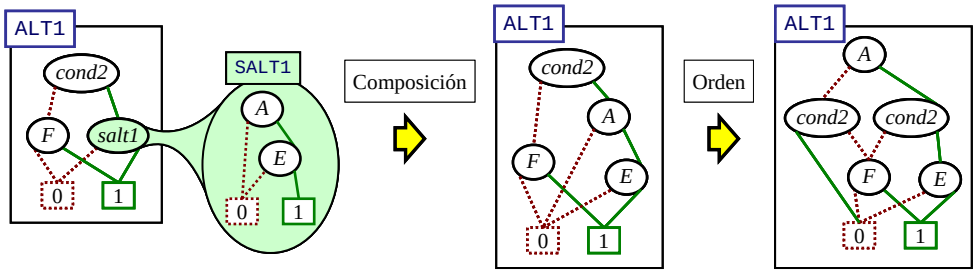


Figura AII-31: Composición de la señal Salt1 en el BDD de Alt1 y cambio de orden del BDD resultante

Nótese que en la figura AII-31 después de la composición se ha cambiado el orden del BDD resultante, poniendo la variable *A* en primer lugar. El BDD resultante de este cambio de orden es mayor, sin embargo se pone la variable *A* primero porque se toma un orden de variables que sigue el recorrido del EHM, y debido a que la variable *A* aparece en *Cond1* su orden es anterior a *Cond2*. Como se explica en el apartado 3.6, seguir un orden así suele ser más ventajoso para el BDD completo de la variable *Z*, aunque no lo sea para señales particulares como *Alt1*. Y así lo es para este ejemplo, en el que el BDD de *Z* es menor colocando la variable *A* primera.

Finalmente, en la figura AII-32 se esquematizan las distintas composiciones de BDD realizadas para formar el BDD de la variable *Z*, que es el utilizado para el cálculo de la probabilidad por el procedimiento explicado en la sección 2.2.2.

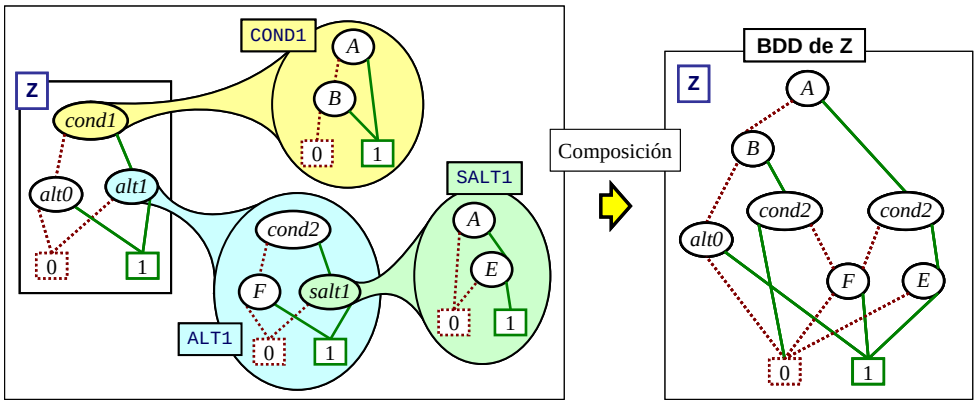


Figura AII-32: Composición de los BDD individuales para formar el BDD de la señal Z

AII.4 aBDD diferenciados e indiferenciados

En esta sección del anexo se explicará con cierto detalle la distinción entre los *a*BDD diferenciados (*da*BDD) y los indiferenciados (*ia*BDD). En muchas ocasiones el *a*BDD resultante es el mismo para ambas representaciones, aunque hay ciertos casos en los que se consigue una importante reducción de tamaño con el uso de *ia*BDD.

Como se ha dicho, en esta tesis no se va a realizar distinción entre ambos y de manera general se denominarán *a*BDD. En el caso de que para la actividad de una señal dichos *a*BDD sean diferentes y no se haya especificado de qué *a*BDD se trata, se supondrá que se refiere al indiferenciado.

En los multiplexores aparecen diferencias de tamaño entre estos *a*BDD, sin embargo esto ocurre sólo a partir de multiplexores de 4 alternativas o más. Las diferencias de tamaños entre estos *a*BDD se estudian en el apartado 3.5.4.3.

En la figura AII-33 se han dibujado los *a*BDD indiferenciado y diferenciado para el multiplexor de 4 a 1. Para este caso, la diferencia de número de nodos no es muy grande: el *ia*BDD tiene 3 nodos menos, lo que es un 11% de reducción. El número de caminos se reduce un 33%.

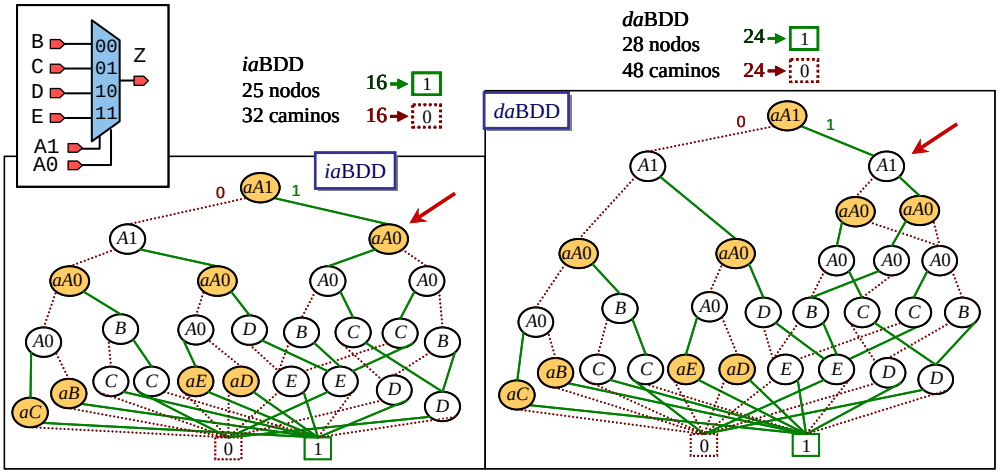


Figura AII-33: aBDD indiferenciado y diferenciado para un multiplexor de cuatro alternativas

Para analizar dónde reside la diferencia entre ambas representaciones, en la figura AII-33 se ha señalado con una flecha el lugar donde los *a*BDD empiezan a ser distintos. La rama cero del nodo raíz a_{A1} es igual en los dos *a*BDD, así que, para facilitar el análisis de las diferencias, se va a sustituir toda esta parte que es igual por una nubecita. Estos *a*BDD que no incluyen la rama cero del nodo raíz a_{A1} se han dibujado en la figura AII-34.

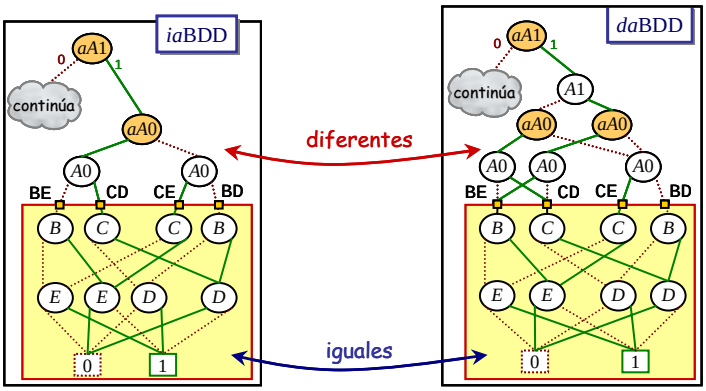


Figura AII-34: Comparación de la zona diferente de los iaBDD y daBDD del multiplexor de cuatro alternativas

Continuando con la figura AII-34, en cada uno de los *a*BDD también se pueden distinguir dos zonas: una que coinciden entre ellas y la otra que para cada *a*BDD es diferente. Las zonas

iguales se han recuadrado, añadiendo una etiqueta a cada una de las cuatro entradas de esta zona. Las etiquetas indican las señales por las que se pasa al recorrer el *a*BDD a partir de dicha etiqueta, estas etiquetas son BE, CD, CE y BD. Éstas se considerarán con un aporte a la ecuación de actividad que es P_{BE}, P_{CD}, P_{CE}, y P_{BD} respectivamente[59].

A la izquierda de la figura AII-35 se ha extraído la parte diferente del *ia*BDD. Ésta se ha recuadrado, y en la parte de abajo del recuadro se han incluido las etiquetas BE, CD, CE y BD, simbolizando todo el recuadro del *ia*BDD de la figura AII-34 que se había indicado que era igual al del *da*BDD.

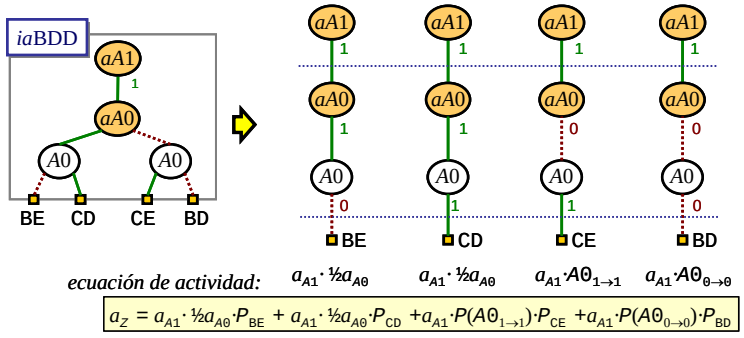


Figura AII-35: Obtención de la ecuación de actividad del iaBDD de la parte diferente al daBDD

A la derecha de la figura AII-35 se muestra el mecanismo de obtención de la ecuación de actividad. Obsérvese cómo al final de cada recorrido figura la etiqueta que le corresponde, y seguidamente aparece la fórmula que representa ese camino.

Extrayendo la parte diferente del *da*BDD se obtiene un recuadro como el mostrado a izquierda de la figura AII-36. Recorriendo sus caminos como se indica a la derecha de la figura se obtienen las transiciones de cada señal. Estas transiciones se han agrupado en aquellos caminos que terminan en la misma etiqueta. El resultado es la misma ecuación de actividad que la de la figura AII-35. Así que con ambos *a*BDD se consigue el mismo resultado numérico para la actividad.

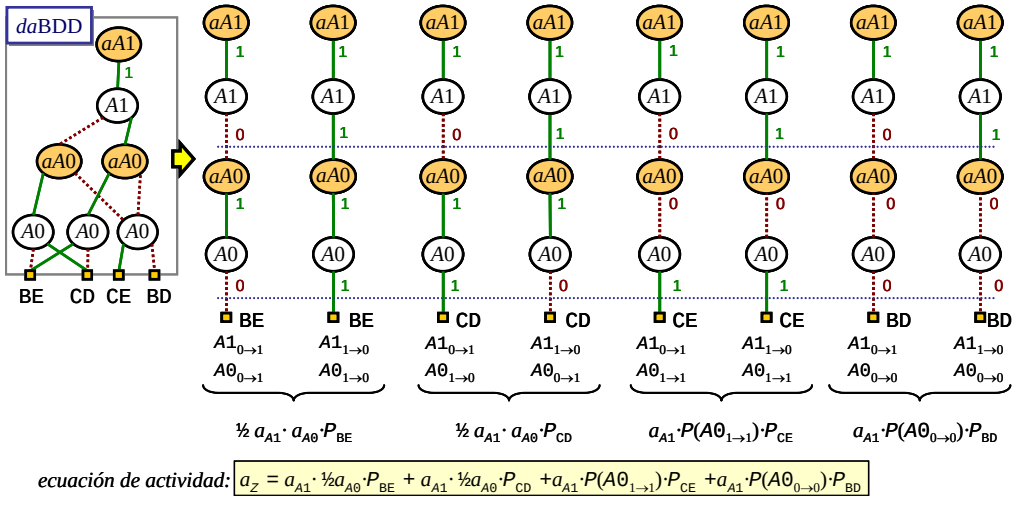


Figura AII-36: Obtención de la ecuación de actividad del daBDD de la parte diferente al iaBDD

Sin embargo, es obvio que el *da*BDD tiene más información que el *ia*BDD, y bajo otras condiciones del modelo (§3.2) esta información puede ser aprovechada. En caso de que se quieran considerar las correlaciones espaciales entre las señales de selección, los *da*BDD proporcionan la información necesaria.

[59] Los valores de estas probabilidades no se han indicado para simplificar la expresión. Sus expresiones son: $P_{BE} = \overline{P_B}\cdot P_E + P_B\cdot \overline{P_E}$; $P_{CD} = \overline{P_C}\cdot P_D + P_C\cdot \overline{P_D}$; $P_{CE} = \overline{P_C}\cdot P_E + P_C\cdot \overline{P_E}$; $P_{BD} = \overline{P_B}\cdot P_D + P_B\cdot \overline{P_D}$

Por tanto, si las señales de selección del multiplexor *A*1 y *A*0 formasen un vector *A*(*A*1,*A*0) con dependencias espaciales entre sus elementos, para hacer el análisis de la actividad se requeriría la información proporcionada por el *da*BDD. En esta situación se tendrían que considerar las transiciones del vector tomadas como un conjunto y no las de sus elementos de manera individual. Consecuentemente, como se muestra en la figura AII-37, habría que evaluar las transiciones agrupadas en un vector, dando como resultado una ecuación con probabilidades de transición de vector: $A_{ij\to jk}$.

En la figura AII-37, después de cada camino se muestra la transición de cada elemento del vector. Por ejemplo, el primer camino implica la transición 0→1 para la señal *A*1 (esto es $A1_{0\to1}$) y la misma transición para *A*0 ($A0_{0\to1}$). Esto implica una transición 00→11 del vector *A* (denominada $A_{00\to11}$), como se muestra con la flecha debajo de cada camino. La evaluación de la probabilidad de cada una de estas transiciones junto con la probabilidad del BDD que hay bajo de ellos (simbolizado por las etiquetas BE, CD, CE y BD) forma la ecuación de actividad que aparece al final de la figura.

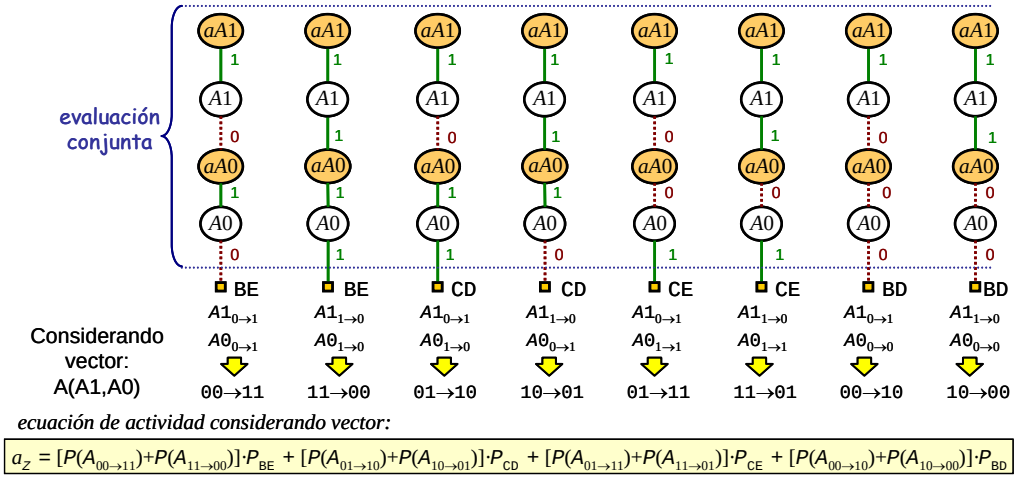


Figura AII-37: Evaluación conjunta del daBDD para la consideración de las correlaciones espaciales de la señal de selección

Sólo si no hubiese correlaciones espaciales, el resultado numérico de la ecuación de la figura AII-37 sería el mismo que el de las ecuaciones de las figuras AII-35 y AII-36. En el modelo de esta tesis (§3.2) las correlaciones espaciales no son consideradas y por tanto se emplearán los *ia*BDD por su menor tamaño. Sin embargo, en caso de que se necesitase considerar estas correlaciones sería preciso emplear los *da*BDD para conservar la información espacial que se pierde en los *ia*BDD.

ANEXO III. ANEXOS A LOS RESULTADOS

En este anexo se amplían los resultados experimentales: analizando con más detalle los modelos de estimación aplicados a los circuitos de pruebas y las causas del error. A partir de la estructura del circuito se observa el cumplimiento de las condiciones que minimizan el error cometido (§3.4.3). Estas condiciones se recuerdan a continuación:

1. Que la señal de selección sea independiente de las alternativas
2. Que la actividad de la señal de selección sea baja
3. Que existe una baja dependencia temporal en las señales reconvergentes
4. Que la relación entre fuentes comunes de las alternativas sea pequeña respecto a las independientes

Por último, para minimizar el error se debe realizar el mínimo de particiones disjuntas, evitando realizar particiones exhaustivas.

AIII.1 Circuito "max"

AIII.1.1 Análisis de los modelos

Como se ha comentado en el apartado 4.2.1, se ha analizado la variación del tamaño de los BDD en función del ancho de bus de los operandos. Para ello se ha modificado el circuito RTL original y se ha sintetizado con el *Design Compiler* de *Synopsys* [125]. La síntesis se ha realizado únicamente con puertas de una y dos entradas. Posteriormente se han comparado los tamaños de los BDD en RTL sin partición por regiones disjuntas con los tamaños de los BDD en puertas. En el apartado 3.6 se explicó la capacidad del nivel RT de lograr un buen orden de las variables del BDD. Por el contrario, en puertas el orden es bastante aleatorio ya que depende de la disposición de las puertas que realice el sintetizador.

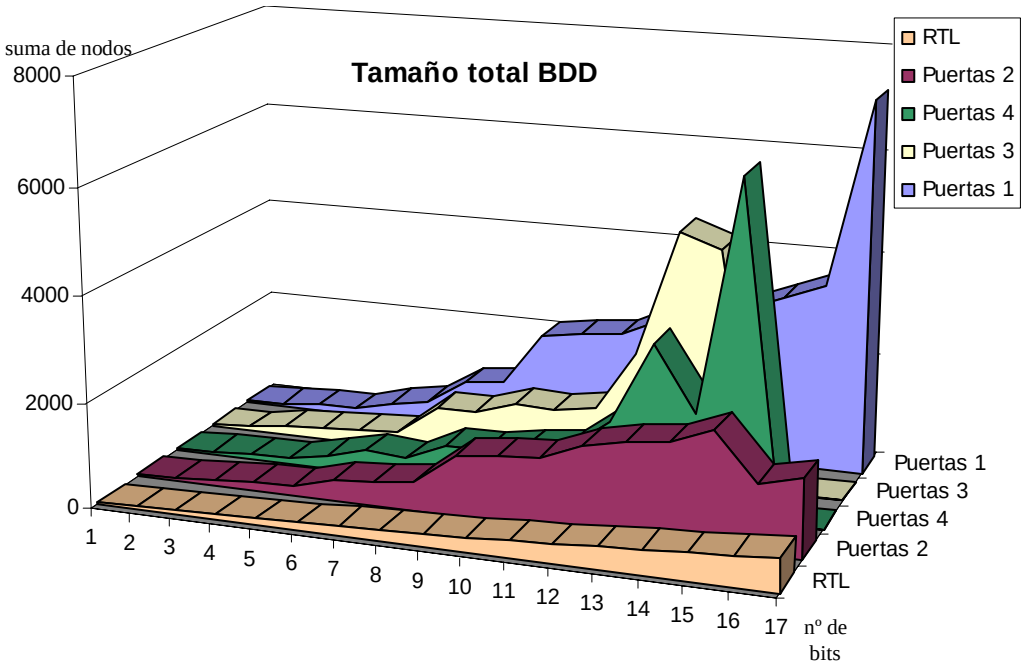
Para demostrar lo aleatorio del ordenamiento en puertas, se han realizado cuatro variantes de cada circuito en puertas, todas ellas equivalentes:

- "Puertas 1": es el circuito en puertas tal como se obtiene de la síntesis
- "Puertas 2": se han cambiado el orden de las entradas de todas las puertas. Si se tenía la sentencia *Z* <= *A* AND *B* se ha cambiado por *Z* <= *B* AND *A*.
- "Puertas 3": se ha cambiado el orden de las entradas de algunas puertas, pero no todas
- "Puertas 4": se ha cambiado el orden de las entradas de aquellas que no se cambiaron en la versión de Puertas 3, dejando el resto como estaban en el circuito original.

Evidentemente, estas modificaciones del orden no afectan al circuito, aunque sí afectan al orden de las variables de los BDD construidos ya que la herramienta de análisis toma el orden de las señales según aparezcan. En puestas, esta manera de ordenar las señales se ha tomado así a falta de unas normas de ordenación como las que se han propuesto para el nivel RTL (§3.6).

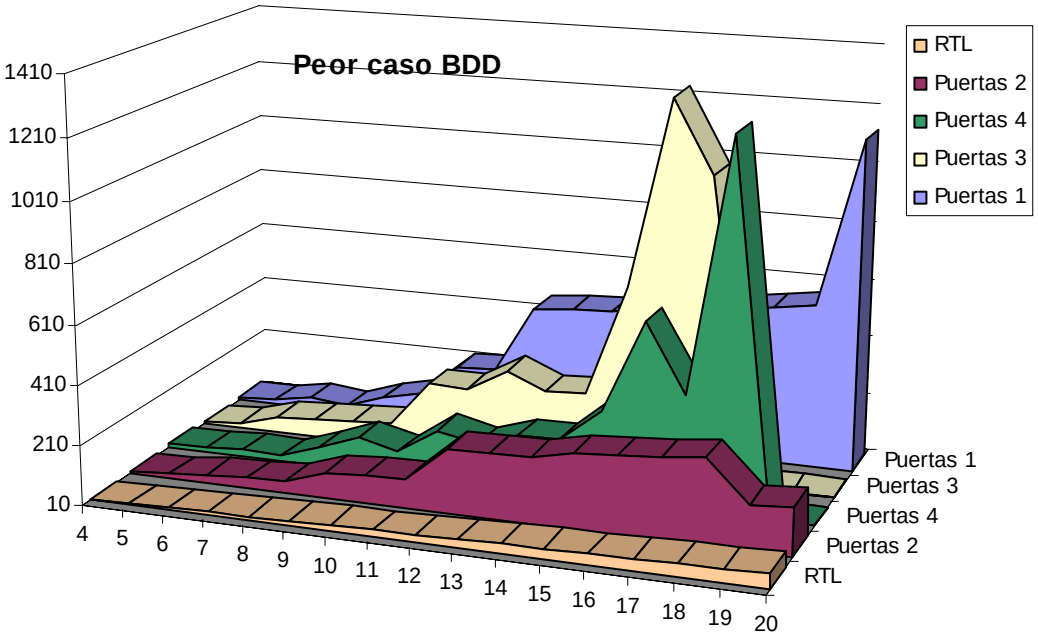
Los resultados numéricos del análisis se han puesto en la tabla AIII-1 (al final de este apartado). A continuación se resumen estos resultados en gráficas.

En la gráfica AIII-1 se muestra la evolución del número de nodos de los BDD de probabilidad con el número de bits. Para la versión "Puertas 3" no se pudo realizar el análisis para 18, 19 y 20 bits por falta de memoria del computador. Por los mismos motivos, para la versión "Puertas 4" tampoco se pudo realizar el análisis para 19 y 20 bits.



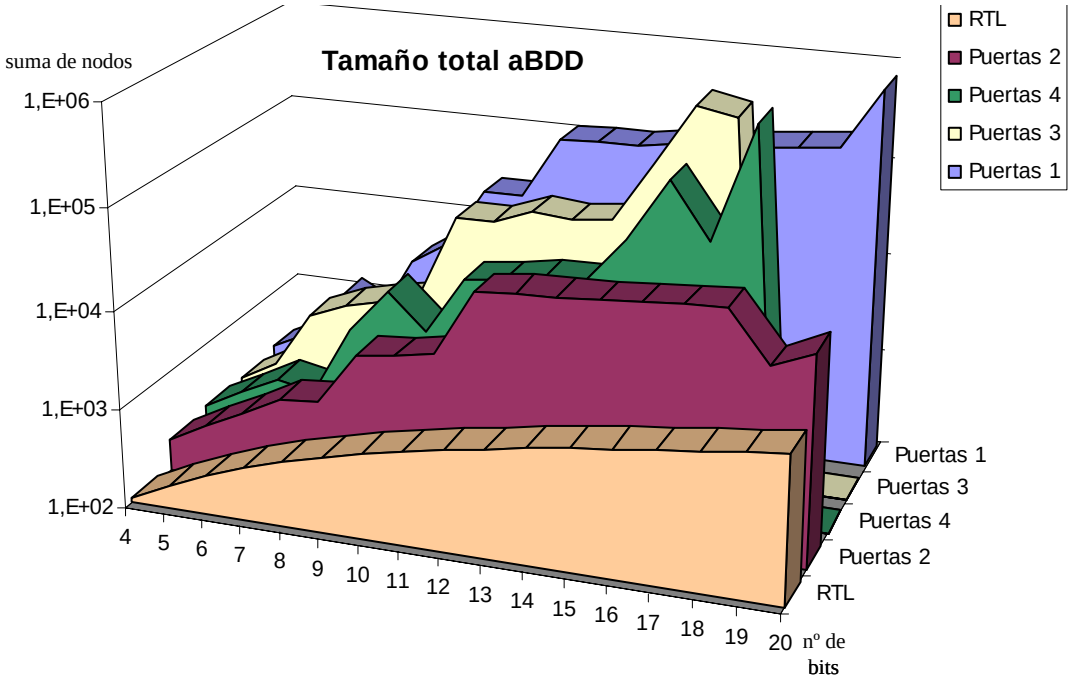
Gráfica AIII-1: Evolución del tamaño de los BDD de probabilidad con el número de bits

En la gráfica AIII-2 se muestra el número de nodos del mayor BDD producido en cada circuito.



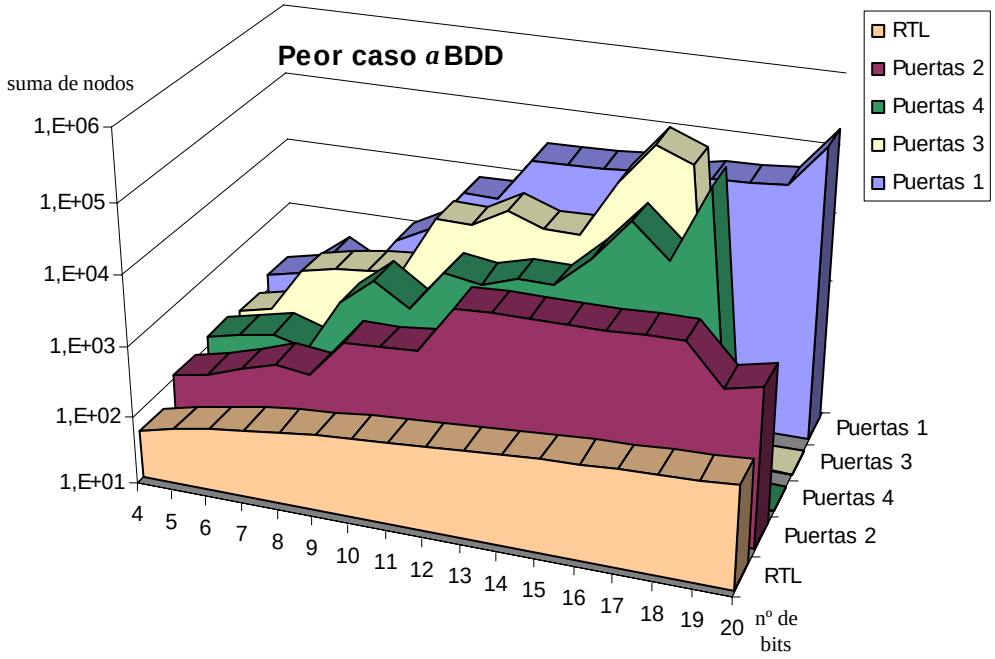
Gráfica AIII-2: Evolución del tamaño del mayor BDD de probabilidad con el número de bits

A continuación se muestra el mismo análisis para los BDD de actividad (*a*BDD). Por haber mayores diferencias de tamaño, las siguientes gráficas se han dibujado a escala logarítmica.



Gráfica AIII-3: Evolución del tamaño de los aBDD con el número de bits

Y a continuación, la evolución del *a*BDD de mayor tamaño con el número de bits (gráfica AIII-4).



Gráfica AIII-4: Evolución del tamaño del mayor aBDD con el número de bits

De las gráficas se puede observar la gran variabilidad de los tamaños de los BDD para los análisis en puertas. Produciéndose a veces grandes diferencias entre los análisis en puertas del mismo número de bits. Incluso, para una misma versión, en algunos casos los BDD son mayores para menor número de bits.

Además, otro inconveniente del nivel de puertas es que los tamaños de los BDD no siguen ninguna secuencia predecible. Dando lugar a grandes oscilaciones de los tamaños de los BDD para diferente ancho de bus.

A continuación se muestran los datos numéricos a partir de los que se han dibujado las gráficas. Aquellas filas que aparece el texto "*sin memo*" indican que no se ha podido realizar el análisis porque el programa se ha quedado sin memoria.

Ancho de bus	Tipo de análisis	Total				Peor caso		
		Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
4	Puertas original	4	74	1118	712	44	861	548
	Puertas 1	5	48	387	260	22	228	154
	Puertas 2	5	46	315	210	18	156	106
	Puertas 3	5	44	331	226	18	172	118
	Puertas 4	5	49	340	237	21	183	131
	RTL	4	29	160	109	12	72	50
5	Puertas 1	7	71	592	404	26	288	198
	Puertas 2	7	74	495	332	23	201	136
	Puertas 3	7	74	551	383	26	210	149
	Puertas 4	7	77	548	372	26	256	174
	RTL	5	44	253	174	15	93	65
6	Puertas 1	9	153	1812	1251	48	672	472
	Puertas 2	9	116	770	514	35	295	195
	Puertas 3	9	181	2125	1413	65	1085	699
	Puertas 4	9	116	886	583	34	344	221
	RTL	6	62	367	254	18	114	80
7	Puertas 1	11	154	1202	803	35	371	244
	Puertas 2	11	161	1227	834	42	414	277
	Puertas 3	11	215	3033	2045	69	1393	916
	Puertas 4	11	113	724	492	29	235	157
	RTL	7	83	502	349	21	135	95
8	Puertas 1	13	333	5732	3747	78	1648	1075
	Puertas 2	13	204	1334	894	47	379	251
	Puertas 3	13	288	3983	2546	81	1691	1046
	Puertas 4	13	276	3860	2536	75	1447	936
	RTL	8	107	658	459	24	156	110
9	Puertas 1	15	478	10345	6423	109	3325	2025
	Puertas 2	15	416	4507	3005	95	1313	860
	Puertas 3	15	331	4866	3127	85	1947	1205
	Puertas 4	15	473	10333	6844	124	3626	2340
	RTL	9	134	835	584	27	177	125
10	Puertas 1	17	966	40400	24919	221	13149	7957
	Puertas 2	17	476	5066	3331	101	1379	896
	Puertas 3	17	896	37030	24656	242	13342	8635
	Puertas 4	17	398	4685	3016	92	1760	1091
	RTL	10	164	1033	724	30	198	140
11	Puertas 1	19	1033	41369	25413	227	13263	8009
	Puertas 2	19	543	5955	3954	106	1538	1004
	Puertas 3	19	922	35414	24011	238	12490	8202
	Puertas 4	19	739	17788	11395	180	7188	4428
	RTL	11	197	1252	879	33	219	155
12	Puertas 1	21	2091	175304	107620	460	56056	33705
	Puertas 2	21	1151	29280	18510	223	7609	4740
	Puertas 3	21	1174	53708	35015	319	24245	15450

	Puertas 4	21	759	20988	12974	153	6211	3756
	RTL	12	233	1492	1049	36	240	170
13	Puertas 1	23	2199	185171	112710	469	58003	34687
	Puertas 2	23	1229	31749	20180	228	8004	5008
	Puertas 3	23	1148	49594	32062	262	16166	10298
	Puertas 4	23	916	24047	15107	195	8635	5240
	RTL	13	272	1753	1234	39	261	185
14	Puertas 1	25	2282	185726	113082	475	58045	34715
	Puertas 2	25	1299	32447	20672	231	8025	5023
	Puertas 3	25	1272	54908	35452	268	16208	10326
	Puertas 4	25	975	24628	15508	198	8656	5255
	RTL	14	314	2035	1434	42	282	200
15	Puertas 1	27	2631	216746	131817	504	64328	38311
	Puertas 2	27	1610	34461	21933	259	8041	5022
	Puertas 3	27	2416	217108	139526	647	101113	64024
	Puertas 4	27	1580	72953	45334	312	23350	14124
	RTL	15	359	2338	1649	45	303	215
16	Puertas 1	29	2873	240378	144624	457	53509	31835
	Puertas 2	29	1787	36476	23262	271	8185	5116
	Puertas 3	29	4829	889787	574773	1288	398138	252214
	Puertas 4	29	3138	303255	187149	621	92603	55912
	RTL	16	407	2662	1879	48	324	230
17	Puertas 1	31	2975	241399	145202	527	68679	40607
	Puertas 2	31	1893	39285	25180	276	8580	5384
	Puertas 3	31	4548	744964	479332	1045	248287	157483
	Puertas 4	31	1911	84378	52903	391	31539	19118
	RTL	17	458	3007	2124	51	345	245
18	Puertas 1	33	3307	267552	161074	548	73240	43237
	Puertas 2	33	2177	40938	26202	295	8473	5304
	Puertas 3	*sin memo.*	-	-	-	-	-	-
	Puertas 4	33	6352	1252325	769300	1248	378508	227927
	RTL	18	512	3373	2384	54	366	260
19	Puertas 1	35	3600	297023	177021	565	77893	45689
	Puertas 2	35	1308	12770	8290	158	2328	1472
	Puertas 3	*sin memo.*	-	-	-	-	-	-
	Puertas 4	*sin memo.*	-	-	-	-	-	-
	RTL	19	569	3760	2659	57	387	275
20	Puertas 1	37	7200	1179494	701609	1133	310509	181869
	Puertas 2	37	1493	20185	12491	167	3257	1986
	Puertas 3	*sin memo.*	-	-	-	-	-	-
	Puertas 4	*sin memo.*	-	-	-	-	-	-
	RTL	20	629	4168	2949	60	408	290

Tabla AIII-1: Tamaños de los BDD del circuito "max" (1) según el análisis realizado

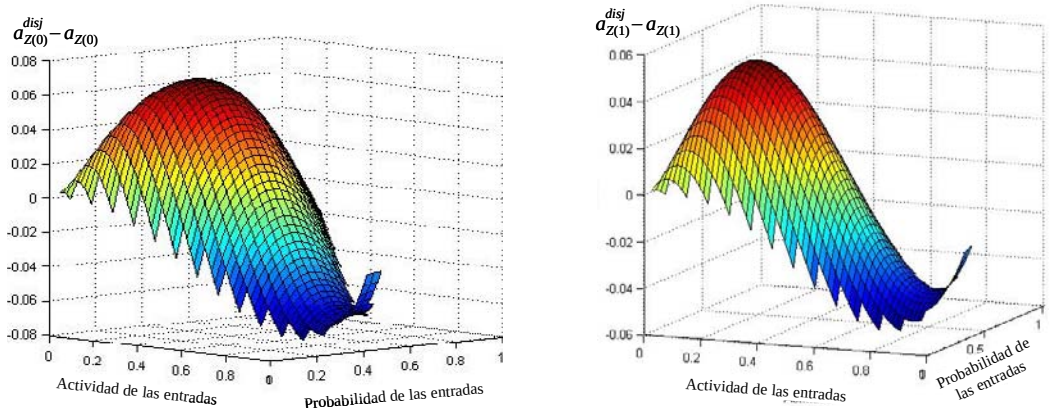
AIII.1.2 Análisis del error

En el apartado 4.3.1.1 se explicaron las razones por las que no conviene dividir el circuito "max" en regiones disjuntas. No obstante, en este apartado del anexo se realizará el análisis del error

debido a la partición por regiones disjuntas para demostrar la validez de las condiciones propuestas en esta tesis para realizar la partición por regiones disjuntas (§3.4.3).

Si al circuito de 4 bits se le realiza una partición en regiones disjuntas exhaustiva[60], de las cuatro señales de salida, hay dos que dan error: *Z*(0) y *Z*(1). Mientras que ni *Z*(2) y ni *Z*(3) dan error, ya que en ellas no se realiza partición.

Los valores de los errores absolutos de la actividad de *Z*(0) y *Z*(1) con partición en regiones disjuntas se han representado en las gráficas AIII-5. En ellas se puede observar el dominio triangular de la función: más ancho para actividades bajas.



Gráfica AIII-5: Distribución del error absoluto de Z(0) y Z(1) para todo el rango de probabilidades y actividades posibles de las entradas (mismo valor para todas las entradas)

En la tabla AIII-2 se muestra el error medio (μ) tomando los errores sin signo y la desviación típica (σ). También se muestra en qué condiciones se dan los errores máximos. Obsérvese que para *Z*(0) y *Z*(1) se dan en las mismas condiciones.

Señales	μ	σ	Error absoluto máximo negativo (e_{max-})					Error absoluto máximo positivo(e_{max+})				
			P_{in}	a_{in}	a^{disj}	a	e_{max-}	P_{in}	a_{in}	a^{disj}	a	e_{max+}
Z(3)	0	0	-	-	-	-	0	-	-	-	-	0
Z(2)	0	0	-	-	-	-	0	-	-	-	-	0
Z(1)	0,024	0,016	0,375	0,75	0,533	0,592	-0,059	0,437	0,203	0,328	0,278	+0,050
Z(0)	0,034	0,020	0,375	0,75	0,497	0,574	-0,077	0,437	0,203	0,348	0,283	+0,065

Tabla AIII-2: Error medio (tomados en valor absoluto), desviación típica, errores máximos y condiciones donde se dan los errores máximos en cada puerto de salida

Los errores mayores se dan en la señal *Z*(0). Para analizar las causas de los errores, en la figura AIII-1 se muestra la partición por regiones disjuntas realizada.

[60] Recuérdese que por partición en regiones disjuntas exhaustiva se entiende la máxima partición que se puede realizar. El resultado es una partición en el mayor número de regiones disjuntas posibles, y por lo tanto, las regiones tendrán un tamaño mínimo (§3.4.3)

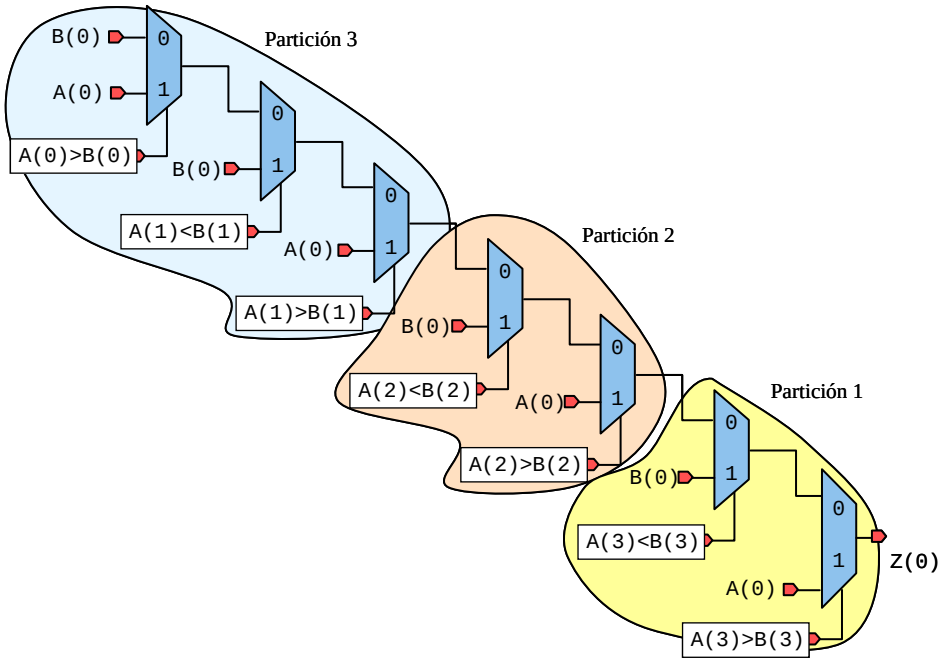


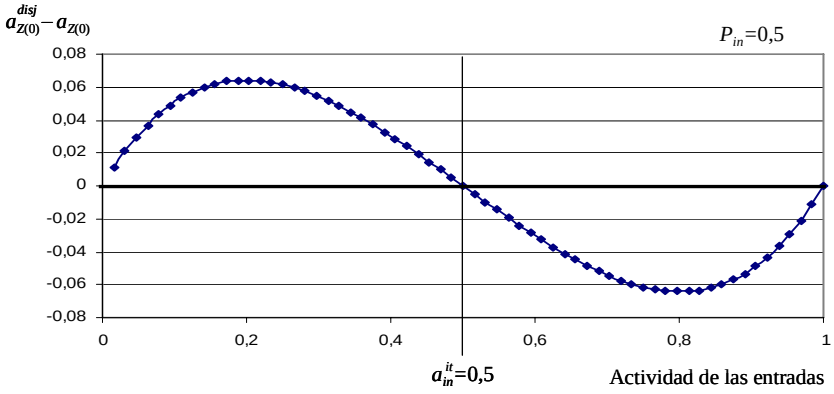
Figura AIII-1: Partición del circuito 1 en RTL para la señal Z*(0)*

Todas las particiones cumplen la condición 1, ya que se había impuesto como obligatoria (§3.4.3). De la figura AIII-1 se ve que las particiones están definidas por las señales de selección. Así, en la partición 1, en las condiciones de los multiplexores están las señales *A*(3) y *B*(3). Para cumplir la condición 1 esta partición no puede dividirse más porque la alternativa cero del primer multiplexor tiene dependencias comunes con la condición.

Se puede observar que todas las alternativas dependen de las señales *A*(0) y *B*(0), siendo estas señales muy reconvergentes, lo que hace que la condición 4 no se cumpla. Ésta es una condición estructural o extrínseca (§3.4.3).

Así pues, a no ser que las señales *A*(0) y *B*(0) sean temporalmente independientes (condición 3) no es recomendable aplicar la partición a este circuito.

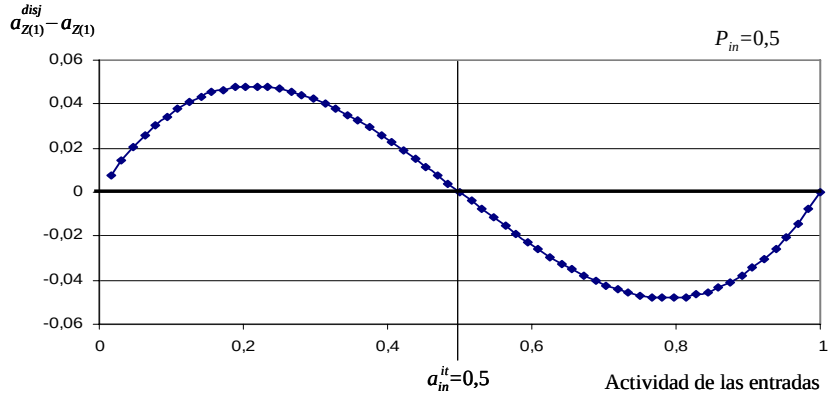
Manteniendo la probabilidad de entrada P_{in}=0,5 y variando la actividad se obtiene la siguiente gráfica de error.



Gráfica AIII-6: Variación del error de $a_{Z(0)}$ *entre el cálculo probabilístico con y sin partición, según se varía la actividad de las entradas y manteniendo la misma probabilidad* (P_{in}=0,5)

En las gráficas, los puntos gruesos indican el valor de la actividad a las entradas para los que se han realizado los cálculos de actividad con y sin partición. Se ha realizado el cálculo de la actividad en 64 puntos, que cubren todo el intervalo de actividades desde *a*=0,015625=1/64 hasta *a*=1, con saltos de la misma distancia. En las gráficas, la línea continua es una interpolación entre los valores experimentales obtenidos. En las ordenadas se muestra la diferencia entre el

valor de actividad para la señal $Z(0)$ con partición: $a^{disj}_{Z(0)}$, y sin partición $a_{Z(0)}$ (y de manera equivalente para $Z(1)$ en la gráfica AIII-7)



Gráfica AIII-7: Variación del error de $a_{Z(1)}$ entre el cálculo probabilístico con y sin partición, según se varía la actividad de las entradas y manteniendo la misma probabilidad (P_{in}=0,5)

A partir de las gráficas se puede observar que cuando a_{in}=0,5 no hay error, ya que se corresponde con el punto donde no hay correlación temporal para la probabilidad dada: como P=0,5 entonces a^{it}=0,5. (§2.2.3.2 ecuación 2.21). Aquellos valores de actividad de las señales reconvergentes $Z(0)$ ó $Z(1)$ menores que a^{it} hacen que se sobreestime el valor de la actividad con partición. Esto es coherente, pues al despreciarse la correlación temporal, el modelo supone que la señal reconvergente tiene una actividad mayor que se propaga a las salidas. Consecuentemente, lo contrario ocurre cuando la actividad de las señales reconvergentes es mayor que a^{it}.

AIII.2 Circuito "comparador"

En la descripción RTL de este circuito, las zonas reconvergentes coinciden con las regiones disjuntas. Por lo tanto, ambas particiones coinciden y no se produce ningún error en la estimación.

La partición para el diseño RTL se muestra en la figura AIII-2, donde se puede ver que las entradas de cada partición son independientes de las entradas del resto de particiones. Y es por esto que coinciden las regiones reconvergentes con las disjuntas.

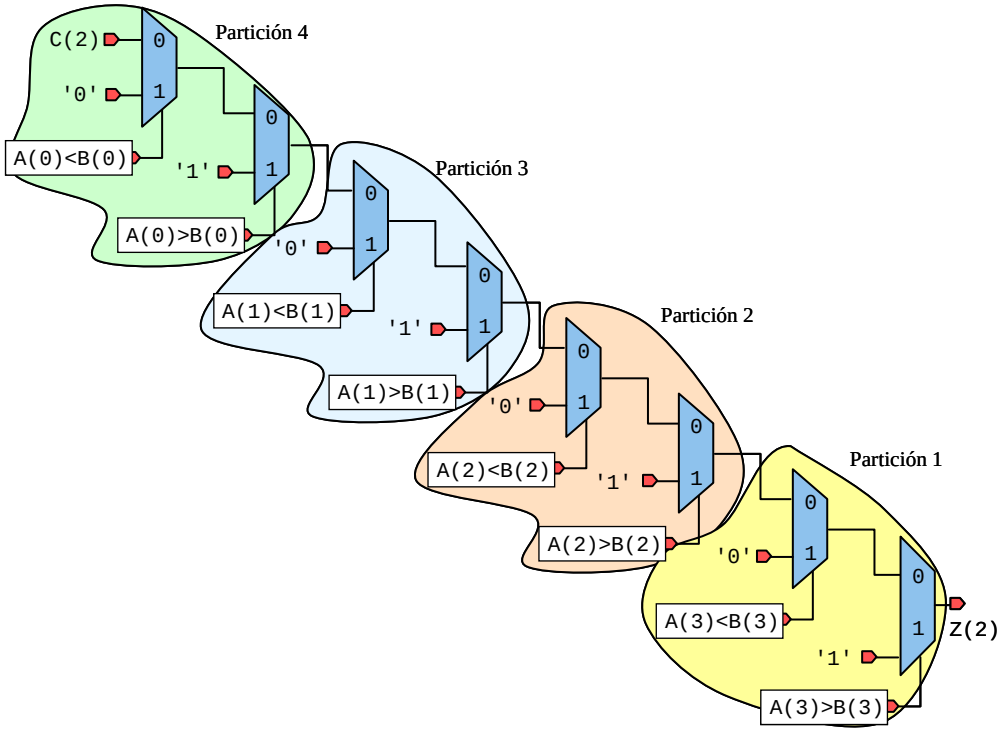


Figura AIII-2: Partición del circuito 2 en RTL para la señal Z(2). Todas las regiones son independientes

A continuación se amplía la tabla 4-3 incluyendo otros órdenes de las variables de los BDD para el circuito en puertas obtenido directamente de la referencia (Navabi [144]), además del obtenido a partir de la síntesis con *Synopsys*.

ancho de bus: 4 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas *Synopsys*	3	51	329	230	19	135	96
Puertas 1	4	46	275	189	16	108	76
Puertas 2	4	49	308	216	19	141	103
Puertas 3	4	55	378	263	19	147	105
RTL sin partición	3	39	222	153	13	81	57
RTL	12	48	204	132	4	18	12

Tabla AIII-3: Tamaños de los BDD del circuito comparador según el análisis realizado

La obtención de los BDD se realiza de manera automática a través de un programa informático. Los órdenes de las variables se construyen según el orden de aparición. Para cambiar el orden, lo único que se ha hecho es modificar las sentencias, por ejemplo, si se tuviese la sentencia:

$$Z <= A \text{ AND } B;$$

Se cambiaría por

$$Z <= B \text{ AND } A;$$

Estas sentencias son equivalentes, y la disposición de las señales depende del sintetizador. Cambiando el orden de algunas sentencias se han obtenido los resultados: "Puertas 2" y "Puertas 3". De la tabla se observa que hay bastante variación de un caso a otro.

Se ha incluido también el resultado del análisis RTL sin realizar partición (ni siquiera por regiones reconvergentes) para poderlo comparar con el resultado en puertas. Ahora el circuito no tiene ninguna partición (tiene tres, una para cada puerto de salida). Y así se ve que ahora el BDD de probabilidad es menor en RTL que en puertas.

Por último, igual que antes, la partición con regiones reconvergentes, es más ventajosa considerando el tamaño de los BDD de actividad, también lo es para el BDD de mayor tamaño.

A continuación se muestran los datos numéricos de la variación del número de nodos de los BDD con el ancho de bus de los operandos.

Ancho de bus	Tipo de análisis	Total				Peor caso		
		Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
4	Puertas	3	51	329	230	19	135	96
	RTL	12	48	204	132	4	8	12
5	Puertas	3	60	478	318	22	202	135
	RTL	15	60	255	165	4	8	12
6	Puertas	3	69	511	346	25	211	144
	RTL	18	72	306	198	4	8	12
7	Puertas	3	175	4500	3063	113	4024	2726
	RTL	21	84	357	231	4	8	12
8	Puertas	3	419	22055	14954	197	10968	7438
	RTL	24	96	408	264	4	8	12
9	Puertas	3	258	6862	4604	115	3364	2258
	RTL	27	108	459	297	4	8	12
10	Puertas	3	195	3696	2460	106	2983	1957
	RTL	30	120	510	330	4	8	12
11	Puertas	3	606	44202	28966	286	22019	14429
	RTL	33	132	561	363	4	8	12
12	Puertas	3	539	30331	19752	251	15076	9817
	RTL	36	144	612	396	4	8	12
13	Puertas	3	276	5563	3690	145	4372	2839
	RTL	39	156	663	429	4	8	12
14	Puertas	3	315	8538	5516	136	4187	2689
	RTL	42	168	714	462	4	8	12
15	Puertas	3	1140	135020	88170	547	67398	44011
	RTL	45	180	765	495	4	8	12
16	Puertas	3	861	81629	53596	406	40695	26719
	RTL	48	192	816	528	4	8	12
17	Puertas	3	2646	824686	523188	1297	412216	261510
	RTL	51	204	867	561	4	8	12
18	Puertas	3	1865	389401	256636	905	194566	128229
	RTL	54	216	918	594	4	8	12
19	Puertas	3	4314	1953470	1259262	2128	976593	629537
	RTL	57	228	969	627	4	8	12
20	Puertas	3	1635	194539	120138	787	97120	59970
	RTL	60	240	1020	660	4	8	12
21	Puertas	*sin memo.*	-	-	-	-	-	-
	RTL	63	252	1071	693	4	8	12
22	Puertas	*sin memo.*	-	-	-	-	-	-
	RTL	66	264	1122	726	4	8	12

Tabla AIII-4: Tamaños de los BDD del circuito "comparador" (circuito 2) según el análisis realizado

AIII.3 Circuito "alu_peq"

La tabla AIII-5 amplía los casos analizados en la tabla 4-4 del apartado 4.2.3. En esta nueva tabla se han añadidos circuitos equivalentes al circuito en puertas original, pero a los que se le han realizado variaciones aleatorias en la disposición de las señales en las expresiones (similar a los realizados en los apartado AIII.1.1 y AIII.2 de este anexo). Todas las variaciones mantienen la funcionalidad original. En el diseño en puertas se puede apreciar que otras disposiciones equivalentes del circuito dan tamaños de BDD diferentes.

	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas Original	17	386	8999	5667	108	3958	2512
Puertas 1	17	382	8879	5574	108	3958	2512
Puertas 2	17	419	10555	6736	123	5407	4016
Puertas 3	17	426	10446	6804	108	3958	2512
Puertas 4	17	409	11145	7000	108	3958	2512
RTL sin partición	7	244	3291	2004	58	689	461
RTL	29	260	2171	1394	55	637	438
RTL-disjunto 1	58	306	1698	981	15	157	85
RTL-disjunto 2	41	274	1638	975	15	157	85
RTL-disjunto 3	31	251	1539	946	15	157	85

Tabla AIII-5: Tamaños de los BDD del circuito "alu_peq" (circuito 3) según el análisis realizado

También se ha añadido el tamaño del RTL sin partición y se puede observar cómo a pesar de que tiene un menor número de particiones, el tamaño de los BDD es menor que en puertas. Usando particiones reconvergentes (fila "RTL") el resultado es todavía más favorable. Recuérdese que estas particiones reconvergentes no producen error.

En la tabla AIII-5 se han incluido tres tipos particiones disjuntas. La primera de ellas es una partición exhaustiva, mientras que la tercera ha realizado un número pequeño de particiones. En el siguiente apartado se analizarán estas particiones disjuntas y su influencia en el error.

AIII.3.1 Partición disjunta exhaustiva y partición disjunta mínima

En la figura AIII-3 se muestra un esquema detallado del circuito "alu_peq".

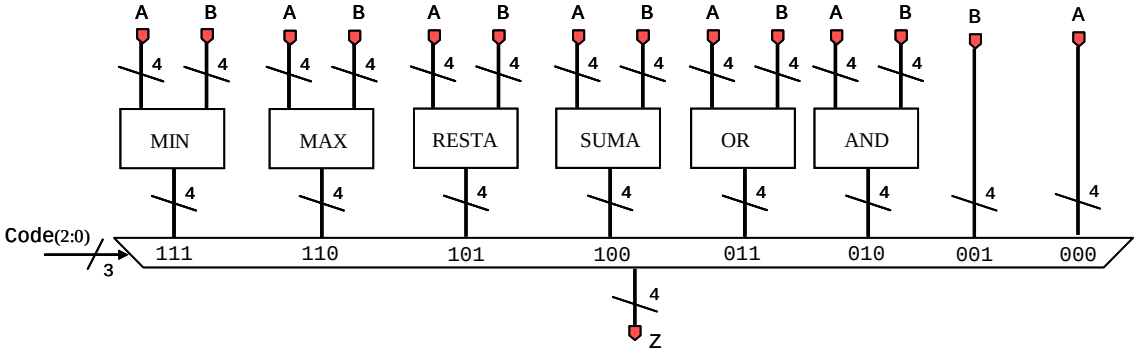


Figura AIII-3: Esquema detallado del circuito "alu_peq"

En la figura AIII-3 se esquematizan todos los bits. Como el cálculo de la actividad se realiza bit a bit, en la siguiente figura (figura AIII-4) se muestra el esquema para el bit 3 (para otro bit el estudio sería similar).

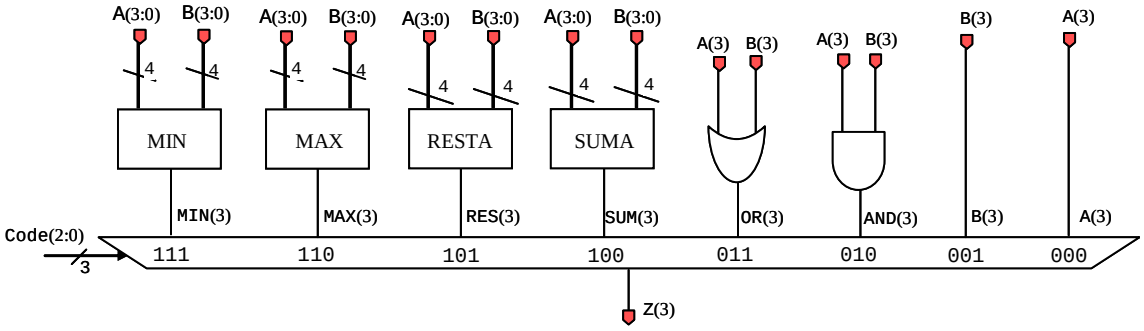


Figura AIII-4: Esquema para Z(3) del circuito "alu_peq"

Las cuatro primeras operaciones ("min", "max", "resta" y "suma") involucran todos los bits, mientras que las cuatro últimas operaciones son operaciones a nivel de bit. Si se hace una partición disjunta exhaustiva, dentro de los bloques "min", "max" también podrían realizarse particiones por regiones disjuntas. Por otro lado, los bloques "resta" y "suma" aceptan partición por regiones reconvergentes (ver apartado el AIII.5.1 del anexo). Recuérdese que esta última partición no produce error en el cálculo.

El esquema de la **partición disjunta exhaustiva** para Z(3) quedaría como se muestra en la figura AIII-5. Esta partición se corresponde con la fila "RTL-disjunto 1" de la tabla AIII-5.

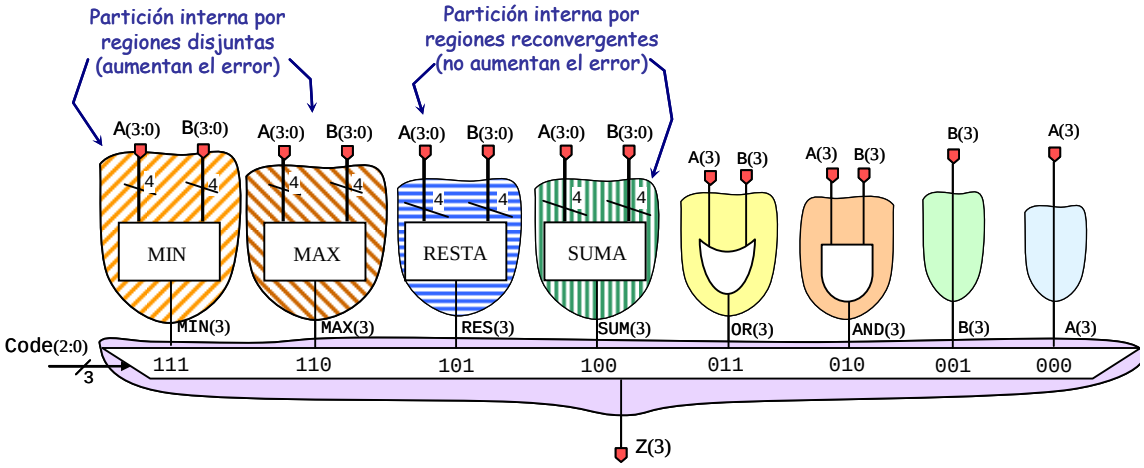


Figura AIII-5: Partición disjunta exhaustiva para el puerto Z(3) del circuito "alu_peq" (RTL-disjunto 1)

Sin embargo, los bloques "max" y "mix" son similares al primer circuito de pruebas (circuito "max", §4.2.1). En el análisis de este circuito se ha visto que el orden proporcionado por el análisis RTL proporciona unos tamaños de BDD manejables. Así que realizar una partición disjunta en este tipo de circuitos aumenta el error de manera innecesaria (ver apartado AIII.1.2). Por tanto, resulta más conveniente no realizar partición en regiones disjuntas para estos bloques.

Para los bloques "resta" y "suma" se puede dejar la partición por regiones reconvergentes ya que éstas no aumentan el error.

La partición para la señal Z(3) quedarían como muestra la figura AIII-6. Esta partición se corresponde con la fila "RTL-disjunto 2" de la tabla AIII-5.

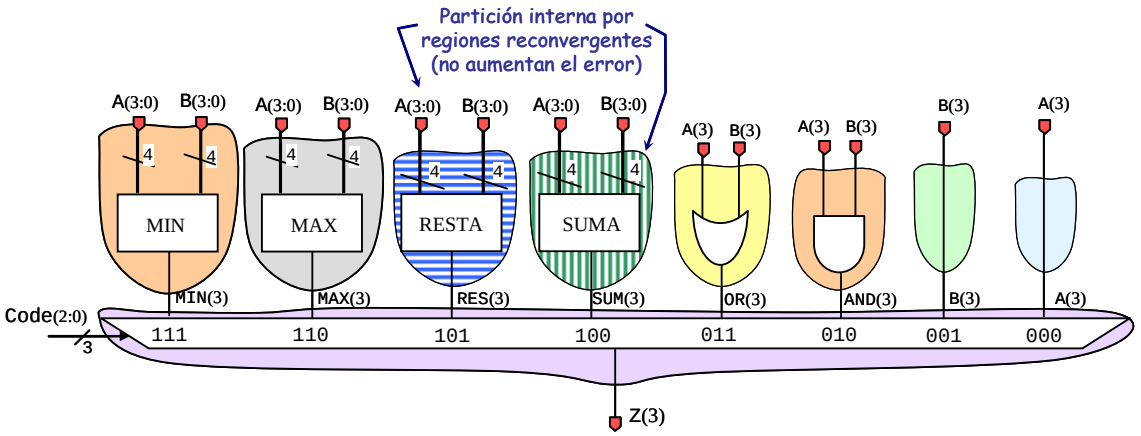


Figura AIII-6: Segunda partición disjunta para el puerto Z(3) del circuito "alu_peq" (RTL-disjunto 2)

De la figura AIII-6 se observa que existe un importante número de particiones disjuntas. Y que hay algunos que son de pequeño tamaño, ya que involucran pocos bits. Estas son las correspondientes a los bloques lógicos: "and", "or" y las que directamente devuelven el operando *A* y el *B*.

Estas regiones se podrían juntar en una sola región. Como sólo involucran a un bit de los operandos, el área total no crecerá excesivamente. La figura AIII-7 muestra cómo quedaría esta partición. Esta partición se corresponde con la fila "RTL-disjunto 3" de la tabla AIII-5. Y esta se denominará **partición disjunta mínima**.

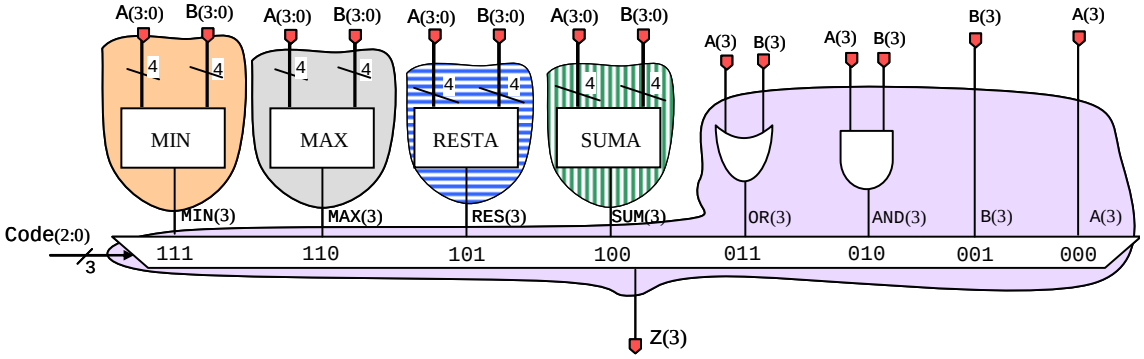


Figura AIII-7: Partición disjunta mínima para el puerto Z(3) del circuito "alu_peq" (RTL-disjunto 3)

En la tabla AIII-5 se ve que la tercera partición es la que menor número de nodos produce, esto se debe a que se eliminan muchas particiones de pequeño tamaño.

En la tabla AIII-6 se muestran los errores medios (tomados en valor absoluto), desviaciones típicas y errores máximos para las tres particiones realizadas. Con el rótulo "disj 1" se designa a la partición disjunta exhaustiva: "RTL-disjunto 1" (figura AIII-5). El rótulo "disj 2" designa a la partición "RTL-disjunto 2", y el rótulo "disj 3" designa "RTL-disjunto 3", que es la partición disjunta mínima.

En la tabla no se han incluido la señal *Flag* por no tener error en los bits 0 y 1, y por su mínimo error del bit 2. De la tabla se puede observar cómo disminuye el error para la partición disjunta mínima ("RTL-disjunto 3").

Señales	μ			σ			e_{max-}			e_{max+}		
	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3
Z(3)	0,021	0,021	0,015	0,019	0,019	0,016	-0,109	-0,109	-0,109	+0,031	+0,031	+0,016
Z(2)	0,023	0,021	0,015	0,019	0,020	0,017	-0,109	-0,109	-0,109	+0,034	+0,032	+0,016
Z(1)	0,023	0,021	0,014	0,020	0,020	0,017	-0,109	-0,109	-0,109	+0,038	+0,035	+0,019
Z(0)	0,024	0,022	0,012	0,021	0,022	0,016	-0,117	-0,117	-0,109	+0,046	+0,042	+0,016

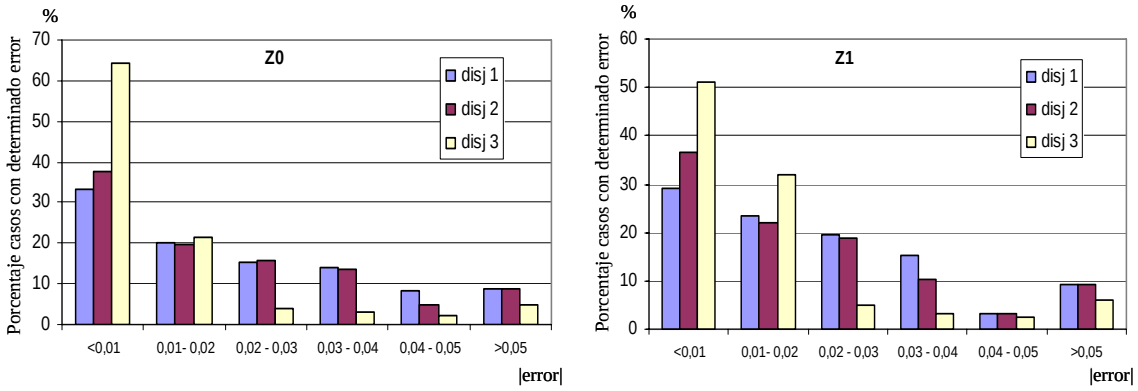
Tabla AIII-6: Error medio (tomados en valor absoluto), desviación típica, y errores máximos para cada puerto de salida y según la partición disjunta realizada

La distribución de los errores también es mucho más favorable para la tercera partición. Ésta se muestra en la tabla AIII-7.

distrib (%)	|e| < 0,01			0,01 < |e| < 0,02			0,02 < |e| < 0,03			0,03 < |e| < 0,04			0,04 < |e| < 0,05			|e| > 0,05		
	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3	disj 1	disj 2	disj 3
Z(3)	32,8	32,8	48,4	24,8	24,8	33,9	24,1	24,1	5,1	5,1	5,1	3,7	3,5	3,5	2,9	9,7	9,7	6,0
Z(2)	27,0	35,1	50,0	25,5	23,3	32,6	21,9	21,7	4,8	12,8	7,1	3,8	3,2	3,1	2,8	9,6	9,7	6,0
Z(1)	29,2	36,6	51,2	23,5	22,0	31,9	19,6	18,9	5,1	15,3	10,2	3,3	3,2	3,2	2,6	9,2	9,1	5,9
Z(0)	33,2	37,8	64,4	20,1	19,5	21,5	15,3	15,6	3,9	14,2	13,4	3,2	8,4	4,9	2,1	8,8	8,8	4,9

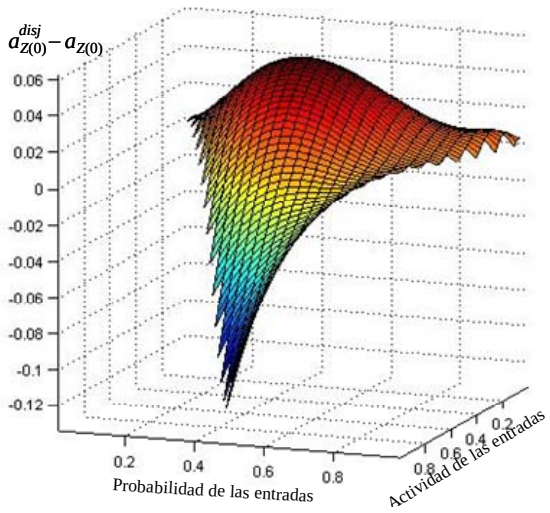
Tabla AIII-7: Distribución del error según el tipo de partición disjunta realizada

Para facilitar la comparación de los datos de la tabla AIII-7, en la gráfica AIII-8 se ha representado la distribución del error para las señales Z(0) y Z(1). En ella se aprecia más claramente la conveniencia de realizar el mínimo de particiones disjuntas ("disj 3"), y evitar realizar particiones exhaustivas ("disj 1").

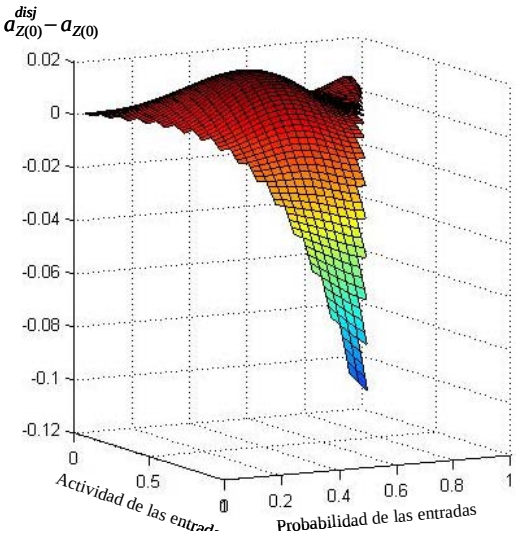


Gráfica AIII-8: Distribución de los errores para Z(0) y Z(1) según el tipo de partición disjunta realizada

En las gráficas AIII-9 y AIII-10 se muestra la distribución del error de Z(0) según los valores de las probabilidades y actividades en las entradas. La gráfica AIII-9 muestra los errores de la partición exhaustiva ("RTL-disjunto 1"), mientras que la gráfica AIII-10 muestra los errores de la partición disjunta mínima ("RTL disjunto 3")



*Gráfica AIII-9: Distribución del error para Z(0) realizando partición disjunta **exhaustiva** (RTL-disjunto 1)*



*Gráfica AIII-10: Distribución del error para Z(0) realizando partición disjunta **mínima** (RTL-disjunto 3)*

A partir de las gráficas AIII-9 y AIII-10, como de la tabla AIII-6, se observa que, si bien realizando la partición disjunta mínima ("RTL-disjunto 3") se reduce considerablemente el error, existen casos donde el error máximo negativo se mantiene elevado. Estos casos se encuentran en la zona de la gráfica donde se asigna una actividad a las entradas cercana a uno. El error máximo se produce en P_{in}=0,5 y a_{in}=1. A continuación se analiza la causa por la que no se reduce el error en estas condiciones.

El máximo error (e_{max-}= -0,109) se produce en el caso extremo, cuando P_{in}=0,5 y a_{in}=1. En estas condiciones de probabilidad y actividad, todos los bits de las señales de selección *Code(i)* cambiarán de valor en cada ciclo del circuito (debido a que *a*=1). Esto provoca que la ALU se quede permanentemente bloqueada en dos operaciones. Las operaciones en las que la ALU se quede bloqueada dependerá del valor inicial de *Code*, habrá cuatro posibles bloqueos:

- 000→111→000→....: Cuando *Code*="000" la salida de la ALU es el operando *A*; cuando *Code*="111" la salida es el resultado del bloque "min".
- 001→110→001→...: Cuando *Code*="001" la salida es el operando *B*; cuando *Code*="110" la salida es el resultado del bloque "max".
- 010→101→010→...: Cuando *Code*="010" la salida es la operación AND; cuando *Code*="101" la salida es el resultado es la resta.
- 011→100→011→...: Cuando *Code*="011" la salida es la operación OR; cuando *Code*="100" la salida es el resultado es la suma.

Con estas condiciones de entrada no hay otras posibilidades de transición. Resulta evidente que estas condiciones **no son realistas con el circuito en funcionamiento normal**, y fue por ello que en el apartado 4.3.1.3 se realizaron análisis limitando la actividad de conmutación de la señal de selección.

En la figura AIII-8 se muestran estas cuatro posibles transiciones. Como se ve, en todos los casos, las transiciones son entre distintas regiones, y es por esto que el error se mantiene con un valor similar al caso de partición exhaustiva.

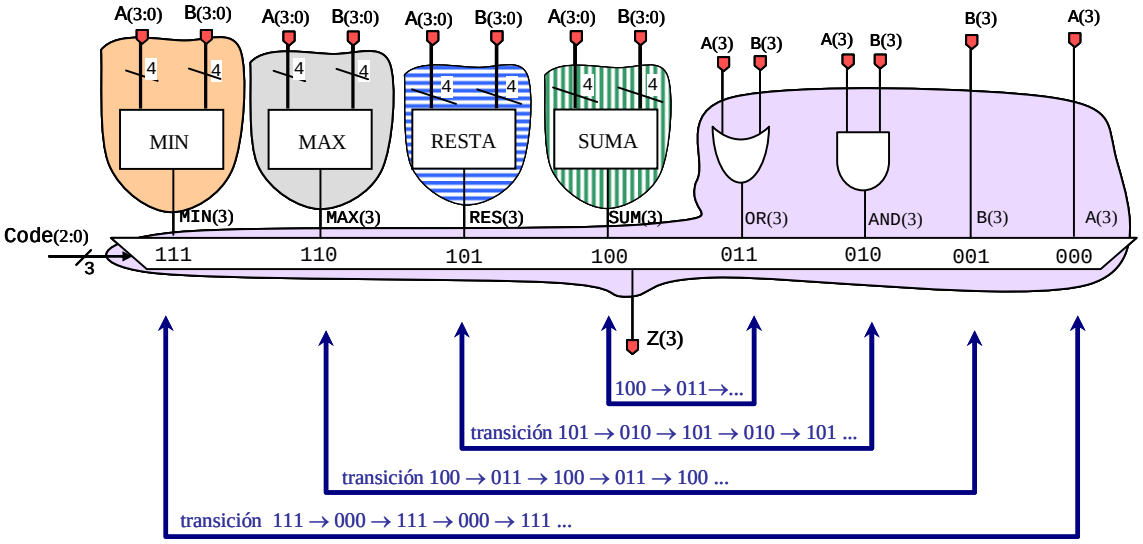


Figura AIII-8: Posibles transiciones cuando $a_{Code(i)}=1$

Por lo tanto, la existencia de este error no es grave, ya que se trata de condiciones irreales. Si, para evitar el análisis en estas condiciones irreales, se **limita la actividad de conmutación de la señal de selección**, el error disminuye en todos los tipos de particiones disjuntas. Aunque es en la partición disjunta mínima ("RTL-disjunto 3") donde los errores resultan menores.

En la tabla AIII-8 se comparan el error medio tomado en valor absoluto, la desviación típica y los errores máximos negativo y positivo, para la partición disjunta exhaustiva ("RTL-disjunto 1") y la partición disjunta mínima ("RTL-disjunto 3").

Señales	μ		σ		e_{max-}		e_{max+}	
	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima
Z(3)	0,0094	0,0051	0,0071	0,0037	-0,039	-0,021	+0,022	+0,012
Z(2)	0,0131	0,0050	0,0085	0,0037	-0,039	-0,021	+0,025	+0,013
Z(1)	0,0136	0,0049	0,0090	0,0039	-0,039	-0,020	+0,029	+0,015
Z(0)	0,0138	0,0037	0,0094	0,0033	-0,030	-0,019	+0,036	+0,010

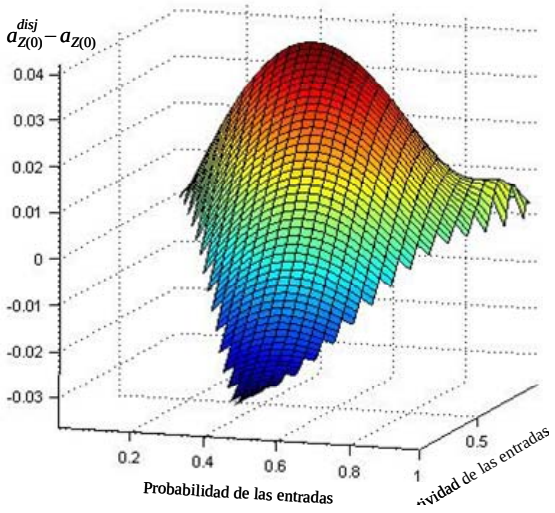
Tabla AIII-8: Error medio (tomados en valor absoluto), desviación típica, y errores máximos para cada puerto de salida y según la partición disjunta realizada. La actividad de las señales de selección está limitada a 0,1.

La distribución del error es muy favorable para la partición disjunta mínima ("RTL-disjunto 3"). En la tabla AIII-9 se comparan la distribución del error de la partición disjunta exhaustiva y mínima. Donde en la exhaustiva algo menos del 50% de los casos no tiene error, mientras que en la mínima rondan el 90% de los casos que no tienen error.

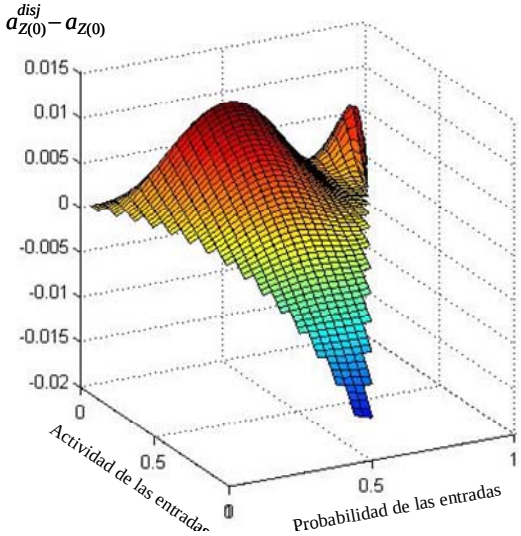
distrib (%)	$\|e\| < 0,01$		$0,01 < \|e\| < 0,02$		$0,02 < \|e\| < 0,03$		$\|e\| > 0,03$	
	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima
Z(3)	59,7	90,3	32,7	9,5	6,2	0,2	1,4	0
Z(2)	40,9	89,9	36,0	10	19,4	0,1	4,0	0
Z(1)	41,4	87,5	32,1	12,4	22,3	0,1	4,2	0
Z(0)	42,5	95,6	29,2	4,4	22,8	0	5,5	0

Tabla AIII-9: Distribución del error según la partición disjunta realizada. La actividad de las señales de selección está limitada a 0,1.

Las gráficas de la distribución del error según los valores de probabilidad y actividad a las entradas, reflejan las ventajas de la partición disjunta mínima. En la gráfica AIII-11 se muestra la distribución del error para la partición disjunta exhaustiva ("RTL-disjunto 1"), y en la gráfica AIII-12 se muestra la distribución para la partición disjunta mínima (RTL-disjunto 3).



*Gráfica AIII-11: Distribución del error para Z(0) realizando partición disjunta **exhaustiva** (RTL-disjunto 1). La actividad de las señales de selección está limitada a 0,1.*



*Gráfica AIII-12: Distribución del error para Z(0) realizando partición disjunta **mínima** (RTL-disjunto 3). La actividad de las señales de selección está limitada a 0,1.*

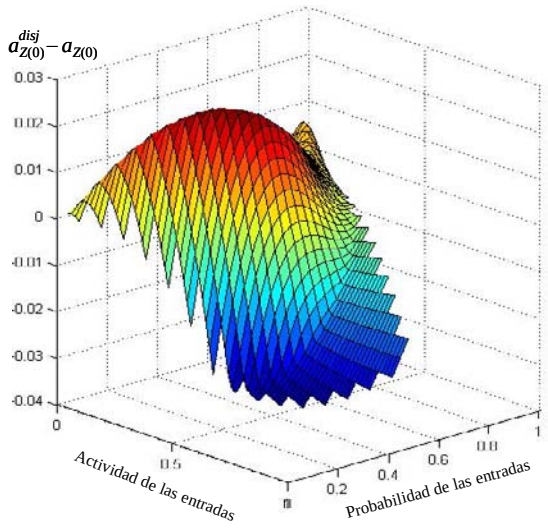
Por último, repitiendo el experimento que se realizó en el apartado 4.2.3 que **limita la correlación temporal** de los operandos, el error sigue siendo menor para la partición disjunta mínima. A continuación se muestran las tablas y gráficas que reflejan las diferencias entre los errores según la partición disjunta realizada.

Señales	μ		σ		e_{max-}		e_{max+}	
	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima
Z(3)	0,0100	0,0065	0,0071	0,0058	-0,023	-0,022	+0,018	+0,011
Z(2)	0,0115	0,0065	0,0067	0,0058	-0,022	-0,022	+0,018	+0,011
Z(1)	0,0116	0,0064	0,0066	0,0058	-0,022	-0,022	+0,019	+0,011
Z(0)	0,0123	0,0057	0,0081	0,0053	-0,035	-0,022	+0,024	+0,010

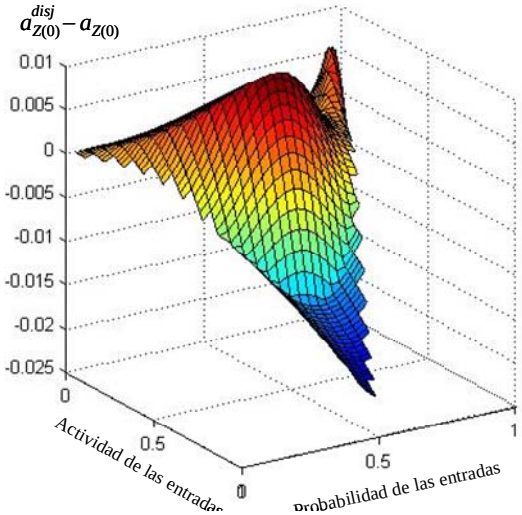
Tabla AIII-10: Error medio (tomados en valor absoluto), desviación típica y errores máximos para cada puerto de salida y según la partición disjunta realizada. La correlación temporal de los operandos está limitada

distrib (%)	$\|e\| < 0,01$		$0,01 < \|e\| < 0,02$		$0,02 < \|e\| < 0,03$		$\|e\| > 0,03$	
	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima	exhaustiva	mínima
Z(3)	56,2	77,2	25,9	19,9	17,9	2,9	0	0
Z(2)	45,4	77,7	36,3	19,6	18,3	2,7	0	0
Z(1)	43,6	78,1	39,0	19,5	17,4	2,4	0	0
Z(0)	46,9	81,0	35,7	18,0	13,3	1,0	4,1	0

Tabla AIII-11: Distribución del error según la partición disjunta realizada. La correlación temporal de los operandos está limitada.



*Gráfica AIII-13: Distribución del error para Z(0) realizando partición disjunta **exhaustiva** (RTL-disjunto 1). La correlación temporal de los operandos está limitada.*



*Gráfica AIII-14: Distribución del error para Z(0) realizando partición disjunta **mínima** (RTL-disjunto 3). La correlación temporal de los operandos está limitada.*

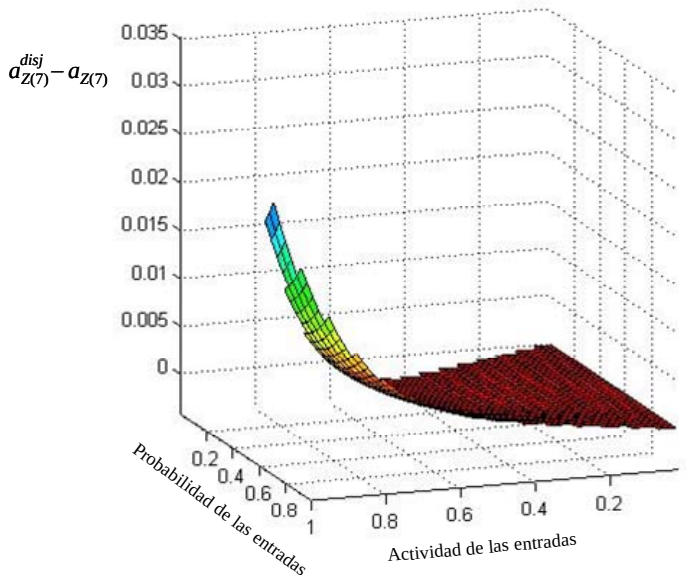
En estas tablas y gráficas se verifica que en todos los casos la partición disjunta mínima produce menos error que la partición disjunta exhaustiva. Se debe tener en cuenta, que en algunos casos de este experimento, la actividad de la señal de selección se mantiene en valores irreales para el circuito en funcionamiento normal: $a_{Code(i)=}1$.

La conclusión de este apartado es que **se debe realizar el mínimo número de particiones por regiones disjuntas de modo que no hagan aumentar los errores en el cálculo, a la vez que limiten el tamaño de los BDD**. Las **particiones disjuntas exhaustivas no proporcionan una clara ventaja en prestaciones** pero **hacen aumentar el error de manera significativa**.

AIII.4 Circuito "alu_core"

Ya se ha comentado en los apartados 4.2.4 y 4.3.1.4 que en este circuito no es necesario realizar la partición por regiones disjuntas. En el apartado 4.3.1.4 se vio que se puede considerar que la partición disjunta mínima no produce error. Ahora, en este apartado del anexo, se analiza el error de la partición disjunta exhaustiva. La partición disjunta exhaustiva no conviene realizarla ya que la partición disjunta mínima produce menor error. Por tanto, este apartado se incluye únicamente para comparar los errores entre particiones disjuntas exhaustivas y mínimas, ya que este error no se dará en la práctica, puesto que se evitará realizar este tipo de particiones.

En la partición disjunta exhaustiva de este circuito, los puertos *Co* no tienen error, mientras que para los bits de la señal *Z* generalmente el error no es importante, y sólo lo es para ciertas condiciones. La distribución del error para la señal *Z*(7) se muestra en la gráfica AIII-15. Para el resto de los bits de *Z* las gráficas resultantes son similares.



Gráfica AIII-15: Distribución del error absoluto de Z(7) del circuito 4 (circuito "alu_core") para todo el rango de probabilidades y actividades

Los errores medios y las desviaciones típicas de cada señal se muestran en la tabla AIII-12. Para el cálculo de los errores medios se han tomado los errores sin signo. Los errores máximos para cada uno de los puertos de salida también se muestran en la tabla. En ella se puede observar que el error medio es muy bajo (0,001). Y que el error máximo negativo no es significativo, mientras que el error máximo positivo se da en condiciones atípicas (P_{in} = 0,5 ; a_{in} = 1).

Señales	μ	σ	Error absoluto máximo negativo (e_{max-})					Error absoluto máximo positivo(e_{max+})				
			P_{in}	a_{in}	a^{disj}	a	e_{max-}	P_{in}	a_{in}	a^{disj}	a	e_{max+}
Z(7)	0,0011	0,0033	0,625	0,281	0,3887	0,3890	-0,0003	0,5	1	0,437	0,406	0,031
Z(6)	0,0011	0,0033	0,625	0,297	0,3974	0,3977	-0,0003	0,5	1	0,437	0,406	0,031
Z(5)	0,0011	0,0033	0,625	0,297	0,3974	0,3977	-0,0003	0,5	1	0,437	0,406	0,031
Z(4)	0,0011	0,0033	0,625	0,297	0,3974	0,3977	-0,0003	0,5	1	0,437	0,406	0,031
Z(3)	0,0011	0,0033	0,625	0,297	0,3974	0,3977	-0,0003	0,5	1	0,437	0,406	0,031
Z(2)	0,0011	0,0033	0,625	0,297	0,3974	0,3977	-0,0003	0,5	1	0,437	0,406	0,031
Z(1)	0,0011	0,0033	0,625	0,297	0,3974	0,3977	-0,0003	0,5	1	0,437	0,406	0,031
Z(0)	0,0011	0,0033	0,625	0,297	0,3974	0,3977	-0,0003	0,5	1	0,374	0,343	0,031
Co(1)	0	0	-	-	-	-	0	-	-	-	-	0
Co(0)	0	0	-	-	-	-	0	-	-	-	-	0

Tabla AIII-12: Error medio (tomados en valor absoluto), desviación típica y errores máximos para cada puerto de salida y las condiciones de probabilidad y actividad a las entradas que los producen. Las condiciones en las entradas son las mismas para todos los puertos.

En la mayoría de los casos (96,8%) el error se puede considerar despreciable, ya que es menor que 0,01. Sólo en un caso de los 1024 el error es superior a 0,03, que además se corresponde con el caso atípico de a_{in}=1. En la tabla AIII-13 se muestra la distribución de los errores de los bits de la señal *Z*. En esta tabla sólo hay una fila porque todos los bits tienen la misma distribución de error.

distribución %	$\|e\| < 0,01$	$0,01 < \|e\| < 0,02$	$0,02 < \|e\| < 0,03$	$\|e\| > 0,03$
Z(i)	96,8 %	2,5 %	0,6 %	0,1 %

Tabla AIII-13: Distribución de los errores para los bits del puerto Z (igual para todos los bits)

Estos errores son de por sí pequeños, pero si se limitase la actividad de la señal de selección como se hizo en el circuito anterior habría errores aún menores. Ya que es en actividades altas donde hay mayor error.

AIII.5 Circuito "addsub"

Al análisis del circuito de 8 bits de bus se ha añadido otra versión del diseño en puertas que se ha creado con otras opciones de síntesis. En la tabla AIII-14 se muestran los tamaños de los BDD. También se han incluido los resultados en RTL figuraban en la tabla 4-5.

ancho bus: 8 bits	Total				Peor caso		
	Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Puertas 1	13	571	11177	8249	139	3837	2856
Puertas 2	13	593	11157	7118	117	4085	2555
RTL	11	499	3599	2590	89	641	479
RTL-disjunto	45	187	645	375	5	19	13

Tabla AIII-14: Tamaños de los BDD del circuito 5 de 8 bits de datos según el análisis realizado

A continuación se muestran tabulados los tamaños de los BDD de los experimentos relativos a este circuito.

nº bits	Tipo de análisis	Total				Peor caso		
		Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
7 bits	Puertas	12	584	11346	7542	99	2633	1746
	RTL	10	398	2838	2026	77	545	405
	RTL-disjunto	40	166	574	334	5	19	13
8 bits	Puertas	13	571	11177	8249	139	3837	2856
	RTL	11	499	3599	2590	89	641	479
	RTL-disjunto	45	187	645	375	5	19	13
9 bits	Puertas	14	1021	36289	22692	225	15749	9737
	RTL	13	660	4898	3549	101	737	553
	RTL-disjunto	51	211	729	423	5	19	13
10 bits	Puertas	16	1155	32565	20461	213	7639	4775
	RTL	14	785	5851	4261	113	833	627
	RTL-disjunto	56	232	800	464	5	19	13
11 bits	Puertas	17	1750	76536	47625	243	16417	10097
	RTL	15	922	6900	5047	125	929	701
	RTL-disjunto	61	253	871	505	5	19	13
12 bits	Puertas	18	1686	64440	39452	261	19829	11907
	RTL	16	1068	7996	5872	137	1025	775
	RTL-disjunto	66	274	942	546	5	19	13
13 bits	Puertas	19	2691	191911	115635	513	78725	47001
	RTL	18	1304	9968	7338	149	1121	849
	RTL-disjunto	72	298	1026	594	5	19	13
14 bits	Puertas	21	2969	172443	104196	501	39127	23479
	RTL	19	1477	11305	8346	161	1217	923
	RTL-disjunto	77	319	1097	635	5	19	13
15 bits	Puertas	22	4284	389598	233800	531	79969	47649
	RTL	20	1662	12738	9428	173	1313	997
	RTL-disjunto	82	340	1168	676	5	19	13

Tabla AIII-15: Tamaños de los BDD del circuito "addsub" (circuito 5) con diferente ancho de bus de los operandos según el análisis realizado

AIII.5.1 Descripción en RTL de un sumador/restador y su partición

Dentro del nivel RT hay diferentes maneras de describir un circuito sumador/restador. Estas diferentes descripciones implican distintos niveles de abstracción dentro del nivel RT. A continuación se analizarán dos posibles descripciones del sumador/restador en RTL:

- Seleccionando la operación mediante multiplexores
- Realizando el bloque sumador/restador usando un único sumador

Habitualmente, el resultado de la síntesis es similar, ya que el sintetizador lo suele sustituir por el mismo bloque sumador/restador[61].

La **primera descripción** utiliza características de más alto nivel que la segunda. Por ejemplo, usando VHDL y mediante el correcto empleo de tipos (por ejemplo *signed* o *unsigned*) y del rango de los vectores para considerar el desbordamiento, un sumador restador se puede describir como se muestra en la figura AIII-9. En ella se muestra la sentencia condicional que

[61] Así lo hace el *Design Compiler* de *Synopsys*

genera el sumador/restador. La representación directa de este circuito (sin especificar los desbordamientos) está en la derecha de dicha figura.

```
if Addsub = '1' then
  Z <= A + B;
else
  Z <= A - B;
end if;
```

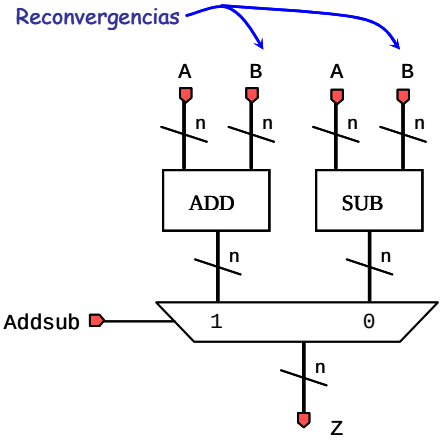


Figura AIII-9: Código que genera el sumador/restador y el esquema RTL directo de dicho código

De la figura AIII-9 se observa que en el diseño hay reconvergencia. Aunque en la figura no se han representado los bits, cada uno de los bits de Z dependen de señales comunes por distintos caminos. Por tanto, con el circuito descrito de esta manera el circuito **no se puede dividir en regiones reconvergentes**, pero **sí en regiones disjuntas**. Este es el circuito se corresponde con el **RTL-disjunto** del análisis (ver tabla 4-6).

Sin embargo, tomando el sumador o el restador de manera aislada, éstos si se pueden partir en regiones reconvergentes. Recuérdese que estas regiones no dan error en la estimación. En la figura AIII-10 se muestra el esquema de un sumador en función de sumadores completos (FA) y un semisumador (HA) para el primer bit. Para este circuito, cada bloque sumador de un bit tiene entradas independientes y por lo tanto se puede dividir en regiones reconvergentes.

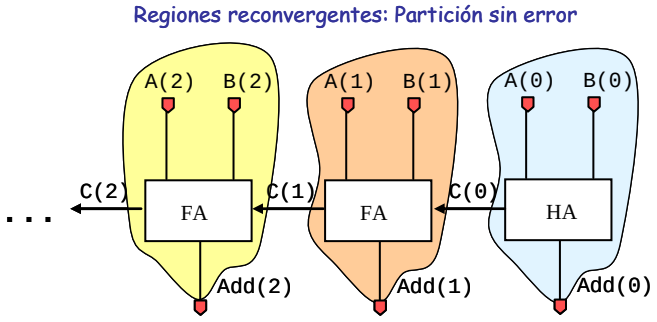


Figura AIII-10: Esquema de un sumador en función de sumadores completos (FA) de un bit y el resultado de su partición en regiones reconvergentes

Para un restador, se puede utilizar un esquema similar al de la figura AIII-10 salvo que emplearían restadores completos y semirrestadores. Y por tanto, para el restador también se podrá realizar la partición en regiones reconvergentes.

Sin embargo, la posibilidad de realizar estas particiones por regiones reconvergentes se pierde al unir el sumador y el restador mediante un multiplexor (figura AIII-9). Y sólo si se realiza la partición por regiones disjuntas en el multiplexor se pueden mantener las particiones por regiones reconvergentes. En la figura AIII-11 se muestra cómo queda la partición para el bit 2 del sumador/restador. En ella, sólo hay una partición por regiones disjuntas, que es la que está en el multiplexor. Gracias a esta partición se pueden mantener las particiones reconvergentes, y con ello, conseguir unos tamaños de BDD pequeños (§4.2.5).

Además, debido a que sólo hay una partición por regiones disjuntas, el error se mantiene bajo. Y lo que es más interesante, este error disminuye conforme aumenta el número de bits, ya que al ser una sola partición, cuanto más grande es el circuito menos influencia tiene su error. Esto se puede apreciar en las tablas 4-16 y 4-17 del apartado 4.3.1.5, en donde sólo para el bit 0 el

error es importante. Como el bit 0 es el que menos área tiene, para este bit se puede dejar sin realizar partición.

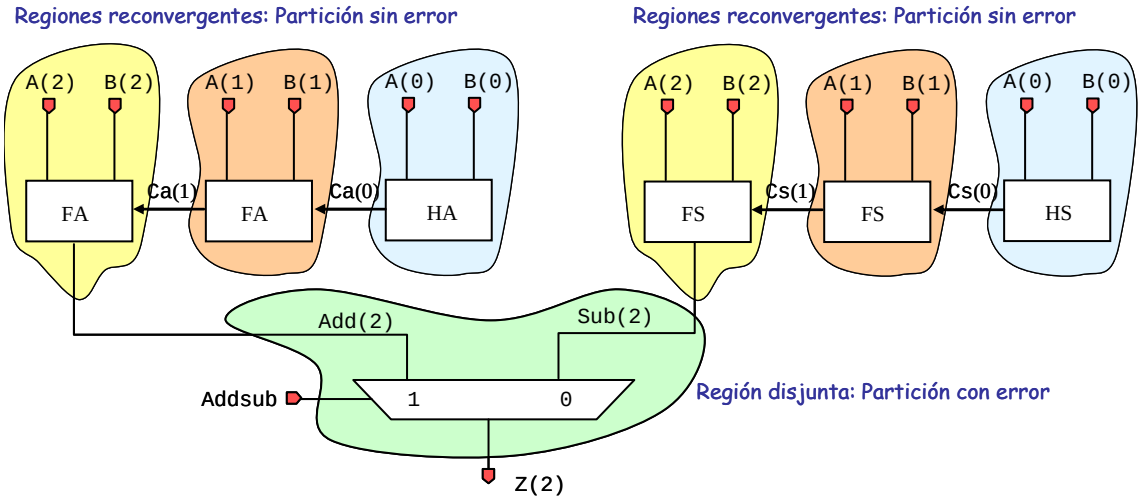


Figura AIII-11: Esquema de la partición del bit 2 del sumador/restador

Otra posible implementación de este circuito es la realizada con un bloque sumador/restador. Este bloque se realiza a partir de un sumador completo, de modo que cuando se resta, se complementa el sustraendo y se pone un uno en el acarreo de entrada para así obtener el complemento a dos del sustraendo. En la figura AIII-12 se muestra el diagrama de bloques de este diseño.

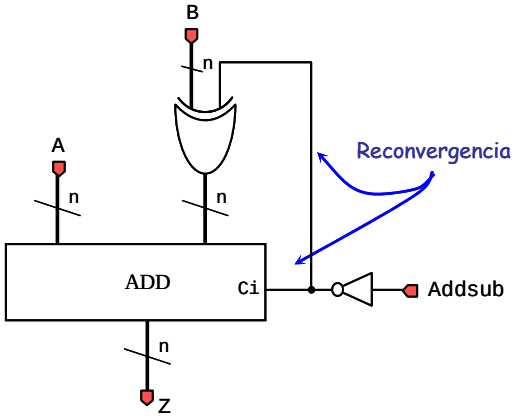


Figura AIII-12: Bloque sumador/restador a partir de un único sumador

El circuito de la figura AIII-12 consume menos recursos que el de la figura AIII-9, ya que tiene un único sumador en vez de un sumador y un restador. Sin embargo, el circuito de la figura AIII-9 es más fácil de entender y está en un nivel de abstracción superior. Por otro lado, descritos desde un lenguaje de descripción de hardware como el VHDL, la síntesis de ambos circuitos da el mismo resultado, ya que el sintetizador transforma el circuito de la figura AIII-9 en el de la figura AIII-12.

Sin embargo, el análisis del circuito de la figura AIII-12 **no permite ninguna partición**, ni siquiera en zonas disjuntas. Esto se debe a que la señal *Addsub* es reconvergente y como ya no hay multiplexor a la salida, no se puede dividir en regiones disjuntas.

Si el circuito sumador/restador estuviese descrito como el de la figura AIII-12, su análisis presentaría los mismos resultados que el análisis RTL del circuito de la figura AIII-9 sin partición por regiones disjuntas. Sin embargo, se podría identificar la estructura de la figura AIII-12 como la de un sumador/restador y realizar el análisis como si fuese el circuito de la figura AIII-9.

AIII.6 Circuito "alu8051_simp"

Los datos numéricos de las gráficas del apartado 4.2.6 se muestran en la siguiente tabla.

nº bits	Tipo de análisis	Total				Peor caso		
		Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
7 bits	Puertas	18	2042	269228	196944	890	233959	174287
	RTL	10	1141	20769	13293	197	4272	2792
	RTL-disjunto	109	682	6314	3700	34	645	370
8 bits	Puertas	21	2198	273365	198584	908	237571	175896
	RTL	11	1423	26061	16813	224	4881	3215
	RTL-disjunto	123	765	7068	4134	34	645	370
9 bits	Puertas	24	2405	112002	70968	408	23641	14967
	RTL	13	1824	33136	21585	251	5490	3638
	RTL-disjunto	138	867	8100	4692	34	645	370
10 bits	Puertas	27	3260	434905	292668	1084	346350	237283
	RTL	14	2172	39658	25971	278	6099	4061
	RTL-disjunto	152	953	8787	5142	34	645	370
11 bits	Puertas	*sin memo.*	-	-	-	-	-	-
	RTL	15	2553	46795	30790	305	6708	4484
	RTL-disjunto	166	1033	9518	5560	34	645	370
12 bits	Puertas	*sin memo.*	-	-	-	-	-	-
	RTL	16	2967	54547	36042	332	7317	4907
	RTL-disjunto	180	1116	10272	5994	34	645	370

Tabla AIII-16: Tamaños de los BDD del circuito 6 según el análisis realizado

AIII.7 Circuito "alu8051"

Los datos numéricos de las gráficas del apartado 4.2.7 se muestran en la siguiente tabla.

nº bits	Tipo de análisis	Total				Peor caso		
		Regiones	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
6 bits	Puertas	9	1721	115902	71589	274	25319	15413
	RTL	9	1763	60051	35278	319	13430	7873
	RTL-disjunto	441	1372	5148	2873	5	19	13
7 bits	Puertas	10	2528	175063	104909	401	44281	25511
	RTL	10	2260	78653	46353	372	15846	9340
	RTL-disjunto	505	1574	5893	3290	5	19	13
8 bits	Puertas	11	4936	760118	467174	1176	275596	166061
	RTL	11	2819	99679	58909	425	18262	10807
	RTL-disjunto	596	1776	6638	3707	5	19	13
9 bits	Puertas	*sin memo.*	-	-	-	-	-	-
	RTL	13	3596	126520	75085	478	20678	12274
	RTL-disjunto	649	2022	7567	4224	5	19	13
10 bits	Puertas	*sin memo.*	-	-	-	-	-	-
	RTL	14	4279	152394	90603	531	23094	13741
	RTL-disjunto	713	2224	8312	4641	5	19	13

Tabla AIII-17: Tamaños de los BDD del circuito 7 según el análisis realizado

AIII.7.1 Descripción en RTL de una ALU y su partición

De manera similar al ejemplo del sumador (ver apartado del anexo AIII.5.1), en RTL hay diferentes maneras de describir una ALU. En la ALU simplificada del ejemplo anterior (ejemplo 6), hay sólo dos operandos. Y por tanto una representación esquemática del circuito podría ser la mostrada en la figura AIII-13. Este esquema es una vista más simplificada de la figura 4-7.

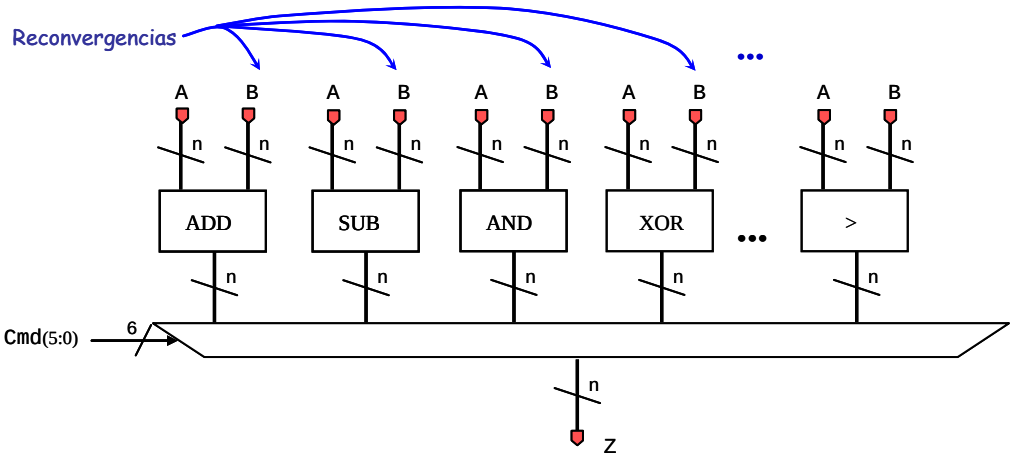


Figura AIII-13: Representación simplificada del circuito 6.

Esta ALU no puede dividirse en regiones reconvergentes debido a la reconvergencia de las señales *A* y *B*. Sin embargo sí se puede dividir en regiones disjuntas, ya que la señal de selección *Cmd* no influye en los datos. Ésta es la partición que se ha realizado para el análisis disjunto del ejemplo 6 (circuito *alu8051_simp*).

Sin embargo para hacer una partición por regiones disjuntas en la ALU completa (circuito 7) se necesita realizar un análisis previo de la ALU (o bien que esté descrita de una más conceptual).

De manera simplificada, la ALU se puede representar como muestra la figura AIII-14. Analizando esta figura se observa que no se puede dividir por regiones disjuntas debido a que las alternativas del multiplexor externo están influenciadas por las condiciones (condición 1 de las reglas para la partición en regiones disjuntas §3.4.3)

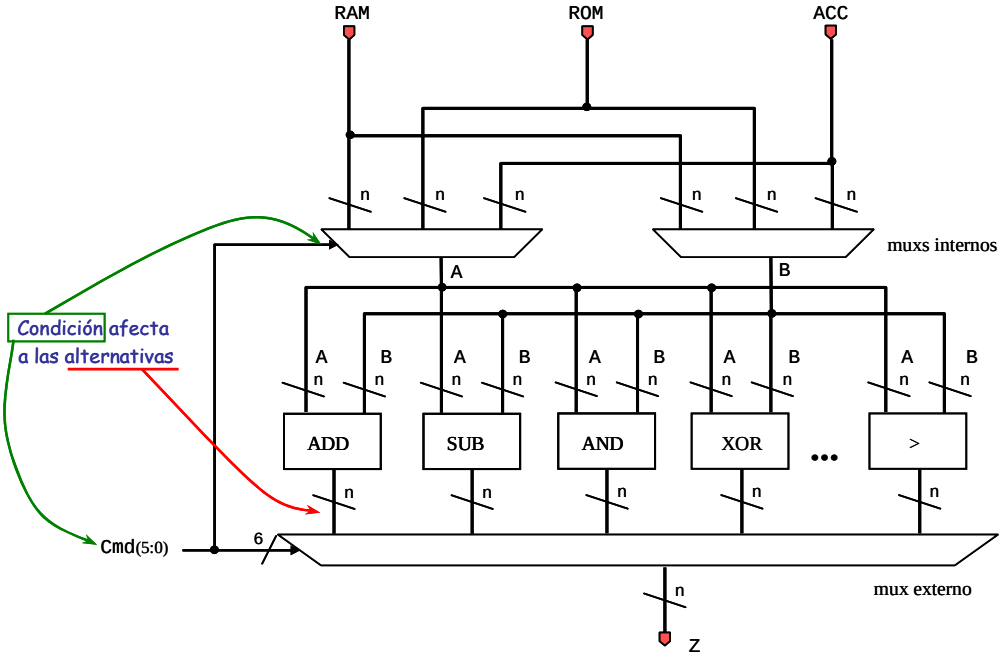


Figura AIII-14: Representación simplificada del circuito 7

Una representación en la que sólo se use un multiplexor externo daría un circuito menos eficiente en área, ya que se repiten cada uno de los bloques que realizan las operaciones (figura AIII-15). Esta representación es conceptual y no sería la que se sintetizase. La ventaja de esta representación es que permite dividir el circuito en regiones disjuntas y se podría obtener a partir del análisis del circuito original de la figura AIII-14. Si bien la representación de la figura AIII-15 permite la división en regiones disjuntas, no se puede dividir en zonas reconvergentes.

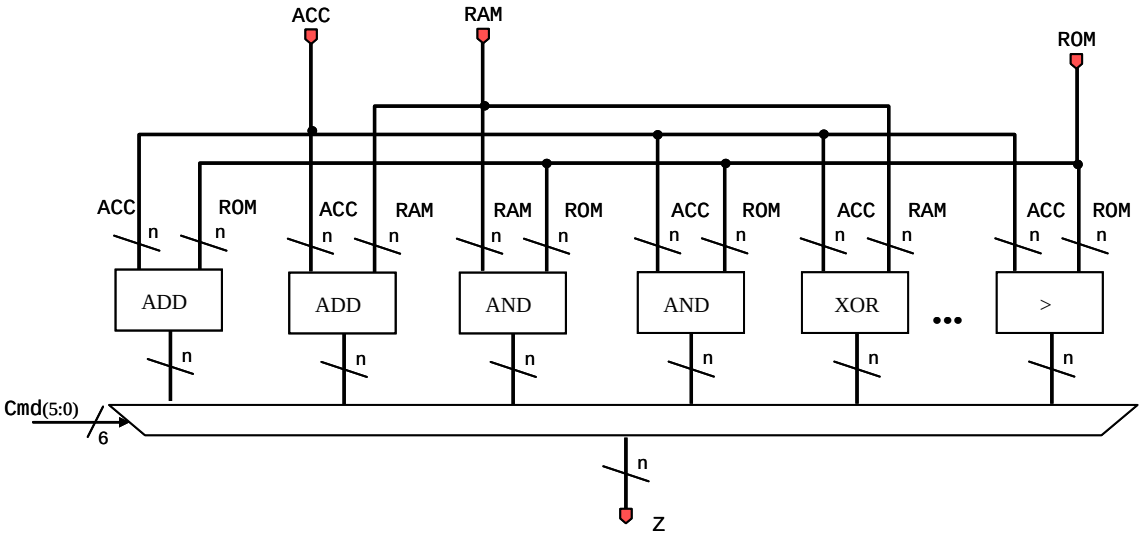


Figura AIII-15: Representación del circuito 7 con un único multiplexor

Gracias a esta representación se ha podido dividir el circuito 7 en zonas disjuntas. Con esto se ha logrado que el aumento del tamaño de los BDD más grandes se mantenga constante (§4.2.7).

AIII.8 Análisis global del errores

En esta sección se incluye la tabla AIII-18 que contiene más información que la tabla 4-28, mostrada en la sección 4.4.5. En esta nueva tabla se incluyen variaciones de los experimentos que evitan condiciones improbables durante el funcionamiento normal del circuito y hacen incrementar el error cometido. Las variaciones de los experimentos se explican en el subapartado del circuito correspondiente, dentro de la sección 4.3.1.

Sólo para el circuito "alu8051_simp" estas variaciones no se llevan a cabo para evitar situaciones anómalas, sino que se llevan a cabo para analizar la actividad cuando se mantiene un operando con una probabilidad y actividad fijas.

Circuito		Peor caso *a*BDD		Estadísticas de los errores			Comentarios / Características del experimento
nº	nombre	RTL-exacto	RTL-disjunto	μ	σ	$\lvert e_{max} \rvert$	
1	max	50	-	0	0	0	sin partición disjunta
2	comparador	12	-	0	0	0	sin partición disjunta
3	alu_peq	438	85	0,0080	0,0143	0,109	situaciones irreales
				0,0027	0,0036	0,021	$a_{Code} < 0,1$
				0,0036	0,0053	0,022	limitada correlación temporal
4	alu_core	168	-	0	0	0	sin partición disjunta
5	addsub	479	13	0,0011	0,0049	0,125	situaciones irreales
				0,0009	0,0031	0,062	$a_{Addsub} < 0,5$
				0,0008	0,0021	0,037	$a_{Addsub} < 0,3$
				0,0006	0,0013	0,012	$a_{Addsub} < 0,1$
6	alu8051_simp	3215	370	0,0011	0,0019	0,012	situaciones irreales
				0,00016	0,00022	0,0014	$P_{A(i)} = 0,5$ y $a_{A(i)} = 0,5$
				0,0011	0,0025	0,029	$P_{A(i)} = 0,5$ y $a_{A(i)} = 0,25$
7	alu8051	10807	912	0,00056	0,00088	0,0068	situaciones irreales

Tabla AIII-18: Reducciones en el tamaño de los aBDD y estadísticas de los errores debidas a la partición disjunta

ANEXO IV. IMPLEMENTACIÓN DEL PROGRAMA DE ESTIMACIÓN

En este apartado se describe brevemente la implementación del programa de estimación de actividad y su integración en la herramienta Ardid, para más información sobre la herramienta Ardid se pueden consultar las referencias [130] y [132].

El programa de estimación de actividad puede ejecutarse de manera independiente por línea de comandos o de manera integrada dentro de la herramienta Ardid. El programa de estimación está escrito en lenguaje C mientras que la integración con el Ardid se realiza tanto en C como en Tcl/Tk.

Las **tareas principales que realiza el programa de estimación** son:

- Tomar el *Modelo Simplificado del Hardware* (SHM [131]) y transformarlo en el *Modelo Extendido del Hardware* (EHM §3.3.2).
- Realizar la partición del circuito en regiones reconvergentes y, si fuese conveniente, en regiones disjuntas (§3.4).
- Construir los BDD de probabilidad y actividad para cada región. Para realizarlo, dentro del programa se ha integrado la biblioteca BuDDy [64]. Esta biblioteca, por ser de código abierto, se ha modificado para incluir la formación y las operaciones relacionadas con los *a*BDD propuestos (§3.5.2). Estos BDD se construyen según el orden RTL propuesto en esta tesis (§3.6).
- Con los BDD construidos se calculan las probabilidades y actividades para dicha señal. El algoritmo utilizado se describe en el apartado AI.2.
- El cálculo de probabilidades y actividades se realiza desde las entradas a las salidas (de menor a mayor profundidad combinacional, §3.3.3). De esta manera, se propagan las probabilidades y actividades de las señales (§3.7).

En realidad, la partición del circuito y la construcción de los BDD no se realizan de una vez, sino que se va realizando a la vez que se propagan las probabilidades y actividades. Esta manera progresiva permite realizar una partición por zonas disjuntas que considere los valores de probabilidad y actividad de las señales y así comprobar dinámicamente si se cumplen las condiciones para la partición por regiones disjuntas (§3.4.3).

La **integración del programa de estimación en la herramienta Ardid** permite proporcionar una información más clara y profunda al diseñador. Lo que es el objetivo último de esta tesis: la ayuda al diseño electrónico. Esta integración facilita el uso del modelo probabilístico propuesto en otros campos de la ayuda al diseño.

Mediante un navegador llamado *Activity Browser* el usuario puede recorrer las regiones en que se ha divido el circuito. En este navegador muestra los valores de actividad y probabilidad de cada señal, y el número de nodos de los BDD producidos para cada región. En la figura AIV-1 se muestra el aspecto de este navegador.

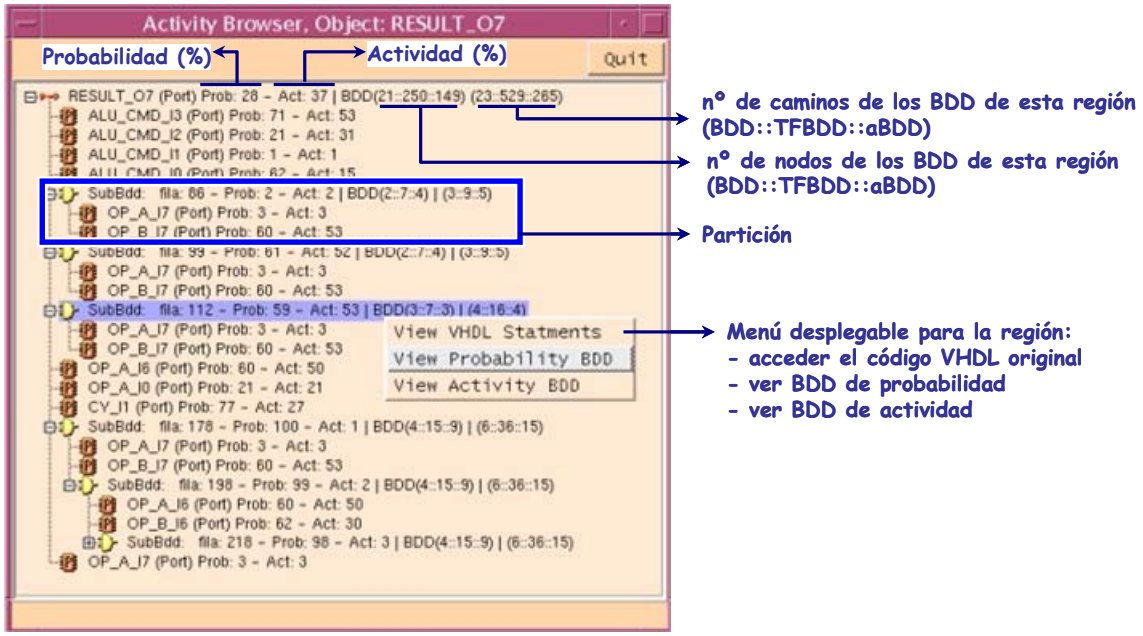


Figura AIV-1: Navegador de actividad (Activity browser)

Como se ve en la figura, desde cada región se puede acceder a un menú desplegable que permite acceder al código VHDL donde se divide cada partición. Este menú también permite la visualización de los BDD generados. La visualización de los BDD tiene meramente fines de depuración ya que para regiones grandes, la visualización de los BDD no resulta práctica. En la figura AIV-2 se muestra la visualización de un *a*BDD de pequeño tamaño: 9 nodos. En ella se muestra de color naranja los nodos de actividad (§3.5.2.1). La visualización se ha realizado integrando la biblioteca de código abierto *Graphviz* [48].

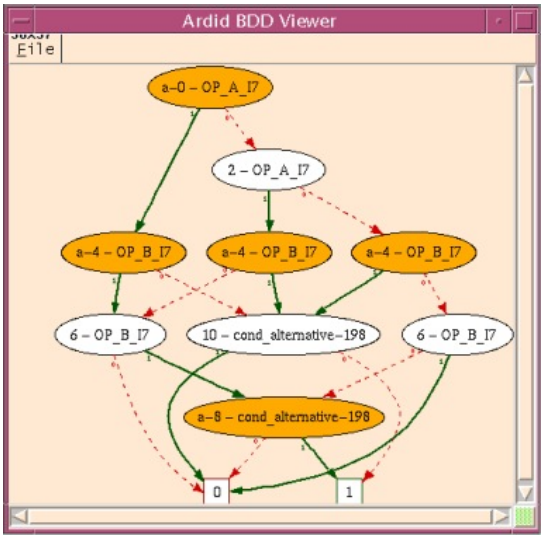


Figura AIV-2: Visualizador de BDD

Por último, de manera preliminar, también se ha implementado una aplicación en la que se puede comparar la actividad media de las señales de cada componente. La figura AIV-3 muestra una imagen de esta aplicación.

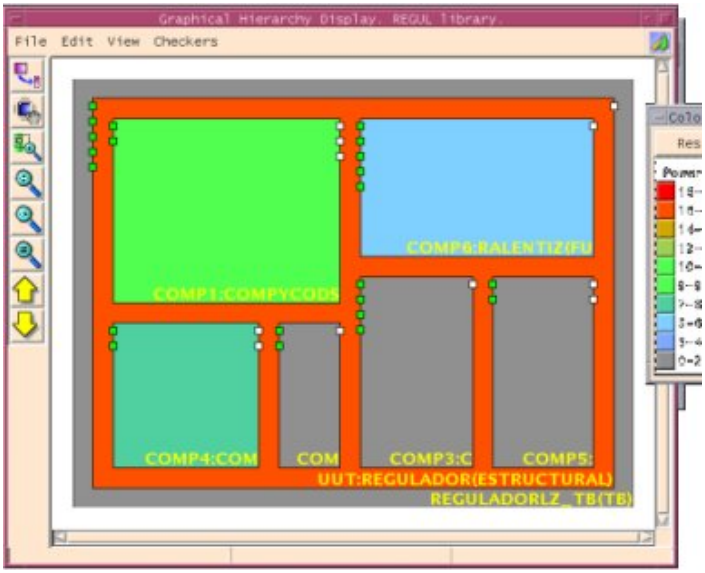


Figura AIV-3: Visualización de la actividad media de los componentes de un diseño

ÍNDICE DE ALFABÉTICO

ENGLISH TRANSLATION

Switching Activity Analysis of Digital Electronic Circuits described at Register Transfer Level using Probabilistic Techniques. Proposal of an Estimation Method

English index

List of figures

List of graphics

List of tables

SUMMARY OF THE THESIS

During the last decades the evolution of digital electronics has been extraordinary. Technology scaling has promoted amazing increments in clock frequency and transistor densities of integrated circuits (IC). As a consequence, circuit functionality and performance has experimented a tremendous growth. Nevertheless, the scaling technologies have confronted designers with new challenges, such as power consumption and reliability. Therefore, nowadays designers not only have to face the functionality of extraordinary complex circuits, but also, they must consider other design issues since the beginning of the design flow.

In addition to these difficulties, time-to-market pressures are aggravated by shorter product cycles due to the rapid technology changes. Nowadays, new product introduction delays can lead to catastrophic consequences, such as lower market share, lower profit margins or the loss of customer goodwill.

To address these challenges, new methodologies and tools for microelectronic design have to be developed. These methodologies and tools should facilitate a seamless design flow in which unnecessary design iterations are avoided. Frequently, despite the circuit functionality is correct, the non-fulfillment of other design issues impel to redesign it.

The ability to estimate the final circuit characteristics constitutes an important aspect of a methodology. Estimators, which anticipate these final characteristics, help designers to make early design decisions and avoid design iterations.

At the present time, power consumption is one of the most relevant issues to be considered in electronic design [63], both due to the proliferation of mobile devices and due to the excessive power consumption of high performance circuits, which require cooling systems that increment the final cost and reduce the reliability. Power consumption is considered one of the main threats for the evolution of semiconductor industry.

Dynamic power consumption, caused by the switching activity of circuit internal signals, represents an important part of the total circuit power consumption. The objective of this Thesis is the analysis of the circuit switching activity, which not only is an essential parameter of the dynamic power consumption, but also may contribute to other analyses such us testability, reliability and quality.

Switching activity estimation techniques can be divided in two broad classes: dynamic and static. Conceptually, dynamic techniques are simpler because they just simulate the circuit under a "typical" input vector sequence and afterwards, the signal activity statistics are collected. The main disadvantages of dynamic techniques are the large simulation times required and their dependence on the input sequence. Dynamic techniques are also known as simulative or statistical techniques.

Static techniques analyze the circuit and elaborate mathematical models that propagate signal probabilities and activities from circuit inputs to outputs. Generally, these models use probabilistic methods; hence, they are also known as probabilistic methods. Static methods are faster than dynamic but their main drawback is the complexity of the resultant model for large designs. In these cases, simplifications, which may decrease the estimation results accuracy, have to be carried out. Static techniques are also known as probabilistic techniques because they usually resort to the theory of probability.

In this Thesis, a probabilistic method for switching activity estimation of register transfer level (RTL) designs is presented. The method elaborates a probabilistic model, which is partitioned to simplify it, and extracts the Reduced Ordered Binary Decision Diagram (ROBDD [16], just BDD henceforth) of the partitions. BDDs are used to calculate signal probabilities and activities, which are propagated through the model. The originality of this work is to take advantage of the RTL descriptions to achieve simpler models by getting better BDD orderings and smaller circuit partitions. Consequently, this Thesis proposes a shift in the probabilistic activity estimation from gate level to RTL, which also have the implicit advantage of being able to

estimate the activity in an earlier design phase. A more efficient representation for activity BDDs is also proposed in this Thesis.

The original Spanish Thesis has been translated to a large extent into English. The first two Spanish chapters have been summarized in one English chapter, which has been called "E1. Previous Work". The English chapter numeration is preceded by an "E" in order to distinguish the English chapters from the Spanish ones. This first English chapter explains the definitions used in the Thesis and the related work. Following to this chapter, the proposed method is explained in chapter E2. This chapter has almost entirely been translated from the spanish chapter 3. This chapter develops the theoretical foundations of the proposed method. Afterwards, chapter E3 expounds the experimental results of the method. This chapter is the translation of nearly the whole Spanish chapter 4. To end, the conclusions of this Thesis are drawn in chapter E4, which is the translation of the Spanish chapter 5.

In addition, the original Spanish Thesis contains four annexes in which supplementary information expands the proposals and the results of the Thesis. These annexes, which have not been translated, are not necessary to understand the main concepts of the Thesis but they could help to understand particular aspects and they also may be useful in order to implement the proposed method. Besides, there are additional results placed in tables and graphs that could be understood by an English speaker even though they are in Spanish.

E1. PREVIOUS WORK

Methods to estimate the switching activity of digital designs are commonly classified in dynamic and static [91], [128]. In dynamic methods, the circuit is stimulated by input vector streams and then, the switching activity statistics of the circuit nodes are collected. Dynamic methods are simple, accurate and general, since issues such as gate delays, glitches and spatial correlations can be automatically considered in the simulation. Their main disadvantage is the large simulation times required to obtain independent results from the input streams, these simulation times may be prohibitive for large circuits. A number of techniques have been proposed in order to reduce these large simulation times, for example: Monte Carlo simulation [18], [141], [27], statistical sampling [34], and vector compaction [77], [81]. Another drawback of dynamic methods is the need to reiterate the entire process when input conditions change. In order to avoid these repetitions, some works propose to simulate the circuit in many different conditions, and then, elaborate an analytical model [12] or a look-up table based model [49].

Static methods calculate switching activity without simulation. These methods elaborate a probabilistic model of the circuit and, by means of that model, signal probabilities and activities are propagated from inputs to outputs. The elaboration of those models requires a one-time intensive circuit analysis, but once the model is built, it is straightforward to obtain the activity results for any other input conditions. The complexity of the model depends on the circuit size and functionality, and also on the model assumptions. Consequently, balance between accuracy and simplicity has to bee found in order to avoid unreliable activity results or excessive usage of memory and computational resources.

To summarize, dynamic methods are easier to implement but they involve extremely large simulation times and the repetition of the computations when input conditions vary. On the other side, static methods implicate the elaboration of a complex model, which, in many times, must be simplified to reduce the computational resources; as a consequence, results may not be always accurate. The model only needs to be elaborated once, even when input conditions vary.

Although no technique is clearly better than the other; nowadays, there is a trend to follow static methods in order to avoid extremely large simulation times [47]. This is one reason why the static method has been chosen in this Thesis. But mainly, the static method has been chosen because the elaboration of the probabilistic model implies a circuit exploration that facilitates other analyses, such as testability, quality and area analyses. The work of this Thesis has taken place in a framework of CAD tools development; consequently, the static circuit analysis benefits from the previous development [131], [132], and will contribute to its improvement.

Prior to summarize the previous work, some of the terminology related to the probabilistic analysis will be defined.

E1.1 Definitions

In this section some of the terminology related to the probabilistic analysis is introduced. The existence of a clock cycle or a minimum time step is assumed; hence, no transitions between two time steps are considered. Therefore, it is a zero-delay analysis in which spurious transitions (glitches) are not taken into account.

- *Signal Probability*, P_x, is the probability that signal x evaluates to logic one. The following representations are equivalent:

  $$P_x \equiv P\{x=1\} \equiv P\{x\}$$

  Whilst $\overline{P}_x$ represents the probability of signal x to be zero. The following representations are equivalent:

  $$\overline{P}_x \equiv P\{x=0\} \equiv P\{\neg x\}$$

Since digital signals are binary, then

$$P_x = 1 - \overline{P}_x$$

- *Strict-Sense Stationary Process* is a process whose statistical properties are invariant to a shift of the time origin. In a combinational circuit, when inputs follows a strict-sense stationary process, their internal nodes also follow a strict-sense stationary process:

$$P_x^{t0} = P_x^{t1} = P_x \quad \text{(E1.1)}$$

Where P_x^{t0} and P_x^{t1} are the probabilities of signal x in time t_0 and t_1, respectively. That is: $P_x^{t0} \equiv P\{x(t_0)\}$.

- *Transition Probability*, $P(x_{i\to j})$, is the probability that signal x experiments a transition from value i to value j (where $i, j \in \{0, 1\}$) in two consecutive time steps. Its equation is:

$$P(x_{i\to j}) \equiv P\{(x^t = i) \wedge (x^{t+T} = j)\}$$

Where $\wedge$ is the operator AND; T is the clock cycle in synchronous circuits or the minimum circuit cycle (no glitches are considered).

Every signal has the same number of transitions from 0 to 1 ($x_{0\to1}$) than from 1 to 0 ($x_{1\to0}$). What can be written as:

$$P(x_{0\to1}) = P(x_{1\to0}) \quad \text{(E1.2)}$$

- *Switching Activity* (or *Signal Activity*), a_x, is the probability that signal x experiences a transition that implies a change of value from a time step to another. Note that this probability definition can only be made in a zero-delay model, where there only can be a maximum of one transition in each time step.

The relationship between transition probabilities and switching activity is:

$$a_x = P(x_{0\to1}) + P(x_{1\to0}) = 2 \cdot P(x_{0\to1}) \quad \text{(E1.3)}$$

- *Signal Inactivity*, $\overline{a}_x$, is the probability of signal x to remain with the same logic value from one time step to another:

$$\overline{a}_x = P(x_{0\to0}) + P(x_{1\to1}) = 1 - a_x \quad \text{(E1.4)}$$

- *Structural dependences*: structural dependences, due reconvergent fanout, constitute an important issue for probability and activity propagation through the circuit. Neglecting these dependences may produce incorrect probability computations. Figure E1-1 shows the effect these dependences may produce on probability computations.

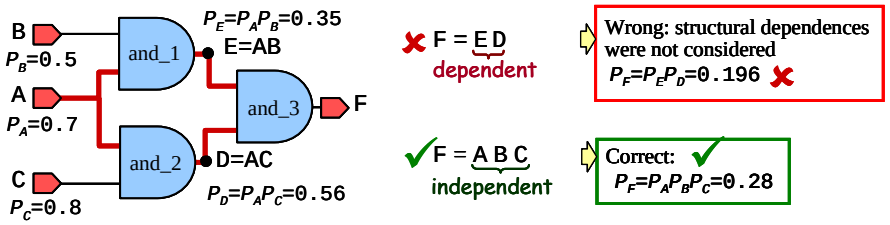


Figure E1-1: Effect of structural dependences on probability computations

To avoid this effect, independent signals have to be searched back and then, calculate probabilities only in terms of independent signals. In the example of figure E1-1, when probability is computed in terms of dependent signals (*D* and *E*) the result is erroneous, but it is correct when independent signals are used (*A*, *B* and *C*).

- *Temporal correlations*: if no temporal correlations are considered, switching activity could be directly computed from signal probability:

$$a_x^{it} = 2\, P_x\, \overline{P}_x \quad \text{(E1.5)}$$

Where *it* stands for *temporal independence*.

However, when the signal value in the next time step is conditioned by its present value (temporarily dependent) there are other possible values for the activity. This fact is observed

in figure E1-2, where signals with the same probability have very different switching activities.

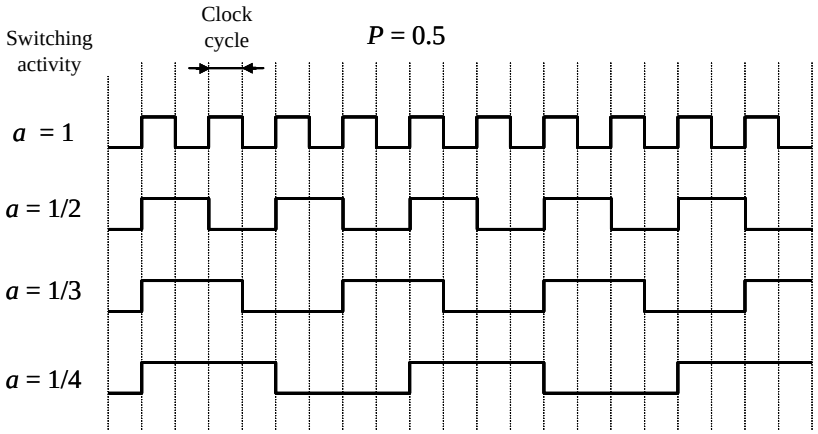


Figure E1-2: Signals with the same probability (P=0.5) but different switching activity

Therefore, equation E1.5 is not always correct; instead, the following inequalities relate signal probability and activity:

$$½\ a_x \leq P_x \leq 1 - ½\ a_x \quad \text{(E1.6)}$$

What can also be written as:

$$a_x \leq 1 - 2 \cdot |P_x - 0.5| \quad \text{(E1.7)}$$

The graphical representation of equation E1.6 is shown in figure E1-3. The area inside the triangle shows a signal's switching activity possible values given its probability value. Note that the curve inside the triangle corresponds to the temporal independence assumption (equation E1.5), which is just a particular case.

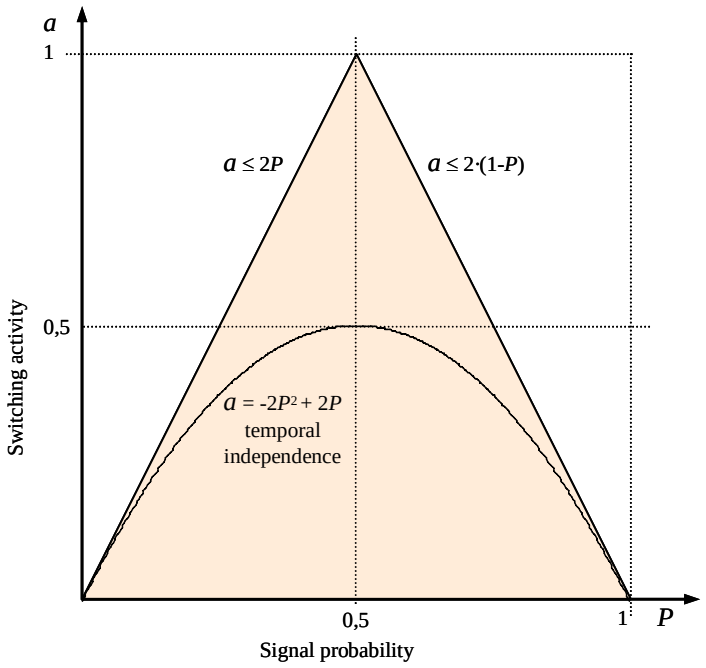


Figure E1-3: Relation between switching activity and signal probability

E1.2 Previous work related to probabilistic estimation

Numerous works have been proposed in the field of switching activity estimation by means of probabilistic methods. The purpose of this section is to present some of the main contributions to this area, but does not intend to be exhaustive.

In the beginning, probabilistic methods arose from signal probability calculations for testability analyses. Signal probability constitutes an essential step to switching activity computation; therefore, many of the proposals related to probability calculations are valid for activity. In fact, signal probability calculation is a particular case of switching activity computation, just as it was exposed in equation E1.5.

One of the earliest works in this field was presented by **Parker** [103]. In this work a polynomial representing each gate output probability is generated (figure E1-4). Afterwards signal probabilities are calculated from inputs to outputs through these polynomials. Structural dependences are considered but, for large circuits, the method could be intractable. This method only calculates signal probability, but not switching activity.

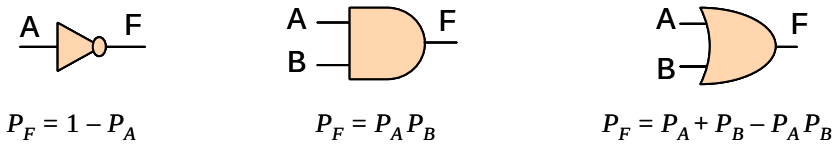


Figure E1-4: Probability equations for some logic functions

In order to reduce the complexity, **Seth** proposed to partition the circuit in *supergates* [121], [122]. Each *supergate* only contains mutually independent signals. Figure E1-5 shows the supergates of an example circuit.

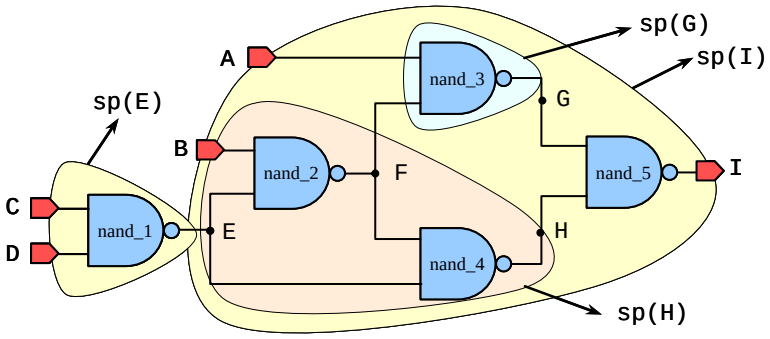


Figure E1-5: Supergates (sp) for an example circuit

Nevertheless, in large and highly reconvergent circuits, supergates may embrace large areas or they may even embrace the whole circuit. In these cases, this work proposed to limit the reconvergence area to a maximum depth from the output of the supergate (see figure E1-6)

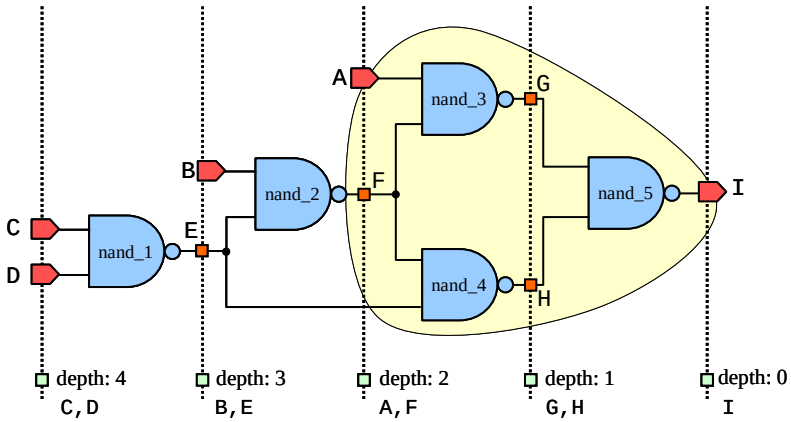


Figure E1-6: Supergate of signal I limited to a depth of 2

The first work that calculated signal activity and power consumption from signal probabilities was proposed by **Cirit** [29]. Since no temporal correlation was assumed in this work, equation E1.5 was used:

$$a_x^{it} = 2\,P_x\,\overline{P}_x = 2\,P_x\,(1-P_x)$$

Ghosh [46] proposed to calculate signal activity performing an XOR at the output function at two consecutive times:

$$a_f = P\{\,f^0 \oplus f^T\,\} \qquad \text{(E1.8)}$$

Probabilities and activities were computed using BDDs, but for large circuits, the method is unfeasible due to the BDD sizes. Hence, approximate simulative techniques were proposed. A method to deal with sequential circuits was also proposed.

Schneider [119] introduced Markov chains, Shannon expansion and reconvergence analysis in order to deal with temporal correlations and structural dependences. In a similar way to Ghosh [46], switching activity was evaluated performing an XOR of the outputs at two consecutive time steps: $Tf = f^0 \oplus f^T$, what was called *Transition Function*. To represent and operate with the transition function, a new BDD representation is proposed: the TFBDD (Transition Function BDD). With reference to the TFBDD variable order, each signal at time 0 (x^0) is followed by itself at the next time step (x^t).

Figure E1-7 shows how TFBDDs are built and how their activity equation is obtained. Similarly to BDDs, TFBDDs are traversed from the root node, A^0, to the terminal node "1". Each path is mutually independent from the rest and, inside each path, only the nodes of the same signal are jointly evaluated.

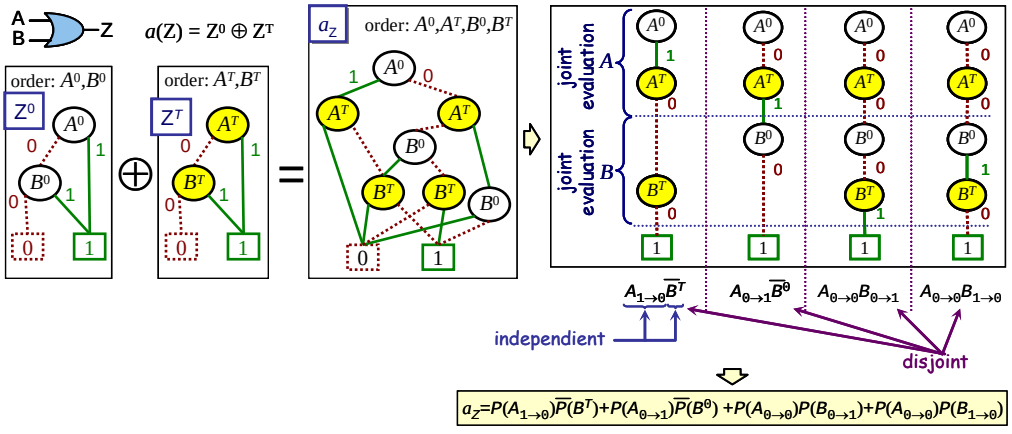


Figure E1-7: TFBDD creation and getting its activity equation for an OR gate

Large circuits were partitioned in reconvergent regions. However, for highly reconvergent circuits, the partition depth should be limited in a similar way as Seth did (figure E1-5).

Marculescu [80] handled input pairwise spatial correlations. BDDs were used to propagate switching activities and spatial correlations. With the purpose of reducing the BDD size, circuit partition dimensions were also controlled by a limit depth.

In order to reduce the size of circuit partitions, **Agrawal** [2] proposed to perform disjoint partitions. At gate level, mutually disjoint signals are not easy to find; therefore, an algorithm to locate them was developed. To determine whether signals are disjoint, a simulation-like procedure was used. Nevertheless, the method does not find all disjoint signals and symbolic analysis was required, resulting in higher time and memory complexities.

Only probabilities were computed in this work, for which the results are exact. However, disjoint regions introduce error for signal activity computation (see section E2.4.3).

In figure E1-8 there is a circuit with disjoint signals. As it can be seen, using disjoint signals, the probability equation results simpler. The result is the same as the one that is calculated by reconvergent regions (independent events). Note that if signals *D* and *E* of figure E1-8 were not mutually disjoint, the result would not be correct, as it happened in figure E1-1.

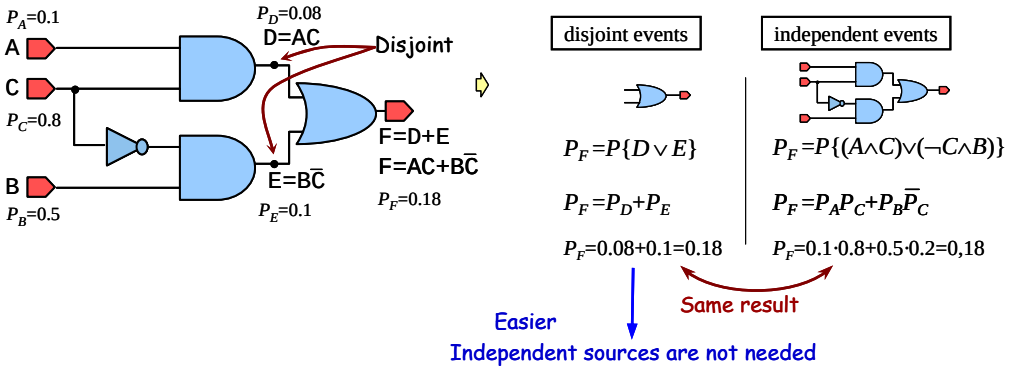


Figure E1-8: Disjoint signals may simplify probability computations

A model proposed by **Theodoridis** [129] suggests the use of *timed based functions* (TBF [73]) to consider real delays in gates and hence, to handle the influence of glitches. The method is computationally intensive; consequently, for large circuits, the model had to be simplified.

Wright [139] estimated switching activity in VHDL combinational circuits, both described in gate level and RTL. The estimation considered neither gate delays nor temporal correlations. The usage of connective BDDs (CBDD [138]) was proposed because they have a linear growth with circuit size. The drawbacks of CBBD include loss of canonicity and, in our opinion, they do not conserve structural dependences.

The method transformed the VHDL design in a collection of boolean equations. These equations were transformed in a gate level netlist in BLIF[62] format, and then, CBDDs were built. In order to reduce computational requirements, CBDDs may also be simplified limiting their depth; as a consequence, the calculation error may increase.

In our opinion, the method does not perform RTL estimation; since it first synthesizes the circuit into gate level, and then it carry out the estimation. Thus, it is performed at gate level.

Bhanja [6] used Bayesian networks to elaborate the probability model. Bayesian networks not only make explicit conditional dependences between nodes, but also they constitute an efficient computational mechanism to update probabilities. Large circuits had to be partitioned to avoid an excessive usage of computational and memory resources. Later, the proposed model was extended to consider input correlations [7] and sequential circuits [5].

In the field of testability analysis, **Fernandes** [38] proposed a probabilistic method to estimate controllability of sequential RTL circuits described in Verilog. The method approximately resolved the Champan-Kolmogorov equations that describe the steady state behavior of the

[62] BLIF: *Berkeley Logic Interchange Format*

circuit. This is the first static approach that calculates probabilities of RTL designs, due to fact that the field of the analysis is testability, switching activities are not computed.

E1.3 Summary of the previous work

Several probabilistic methods have been proposed to compute signal probability and activity. Some of these works have been developed for testability analysis; thus, the computation is restricted to signal probability. Nevertheless, many of the techniques for signal probability computation are valid for activity.

The first methods were simple. Then, they have been extended including more characteristics that made the methods more exact but much complex. Therefore, simplification techniques were needed to cope with the large amount of computational and memory resources needed. On the other side, these simplifications increased the error.

Almost all the methods work at gate level. Wright et al. [139] claim that their method estimated activity of behavioral designs; nevertheless, we consider that they first perform a logic synthesis and then perform the estimation. Therefore, it is not strictly behavioral.

We consider that Fernandes et al [38] is truly the only one that performed the probabilistic estimation at RTL; however, because their method is oriented to testability, they just estimated signal probability and not signal activity.

Table E1-1 summarizes the different proposals. Some of the works in the table have not been described, but the bibliographic reference to them has been included[63]. Cells having the symbol "✓" inside designate that the method considers the aspect indicated in the corresponding column. If the cell has the symbol "~✓" indicates that the method performs an approximate calculation. An empty cell indicates that that aspect is not considered.

	Structural dependences	Activity	Temporal correlations	Simultaneous transitions	Real delays	Spatial correlations at inputs	Memory Elements	RTL
Parker'74 [103]	~✓							
Seth"85 [121]	~✓							
Cirit'87 [29]		✓						
Najm'88 [93]	~✓	✓						
Erolani'89 [37]	~✓					~✓		
Najm'91 [90]	~✓	✓	✓					
Ghosh'92 [46]	~✓	✓	✓	✓	✓		✓	
Tsui '93 [134]	~✓	✓	✓	✓	✓			
Chou'94 [28]	~✓	✓	✓	~✓				
Schneider'94 [118]	~✓	✓	✓	✓				
Marculescu'94 [80]	~✓	✓	✓	✓		~✓		
Tsui'95 [133]	~✓	✓		✓			✓	
Schneider'95 [120]	~✓	✓	✓	✓			✓	
Chou'95 [26]	~✓	✓	✓	✓			✓	
Costa'97 [31]	~✓	✓	✓	✓	✓			
Agrawal"98 [2]	✓							

[63] They have been described in the original Spanish version of the Thesis.

Ferreira'00 [41]	~✓	✓	✓	✓	✓			~✓
Theodoridis'00 [129]	~✓	✓	✓	✓	✓			
Wright'01 [139]	~✓	✓						~✓
Bhanja'01 [6]	~✓	✓	✓	✓				
Bhanja'02 [7]	~✓	✓	✓	✓		✓		
Fernandes'04 [38]							✓	✓
Bhanja'05 [5]	~✓	✓	✓	✓		✓	✓	
Hu'05 [57]	~✓	✓	✓	✓	✓			
This Thesis	✓/~✓	✓	✓	✓				✓

Table E1-1: Summary of the different proposals of probabilistic estimation

E1.4 Thesis framework summary

In this chapter, the two methods used to estimate signal activity has been exposed: dynamic and static methods. Static methods are also known as probabilistic methods because they usually resort to probabilistic techniques.

In this Thesis, the probabilistic method has been chosen due to the smaller computational times required and also to continue the research topics developed in the department.

The work of this Thesis is included in a wider research line which is aimed to develop methods and tools to assist digital designers to improve circuit quality and reduce design time. All these tools are brought together in Ardid [132], which is a front-end environment specially designed to work with VHDL designs at RTL. Ardid elaborates a *Simplified Hardware Model* (SHM [131]), which is a hardware oriented schema of the circuit. Thanks to SHM, circuit analyses are carried out, such as quality [130] and area analysis [76]. An objective of this Thesis is to continue with this research line. And thus, since the probabilistic analysis needs the elaboration of a more detailed hardware model, it would contribute not only to the activity analysis, but also to other new analyses, such as quality [20], [132] and testability analyses [21].

One of the main differences between dynamic and static techniques is that dynamic techniques do not require a deep circuit analysis. Hence, dynamic techniques can be applied to a wide range of description levels.

On the other hand, **static** (or probabilistic) techniques, which are the objective of this Thesis, are conditioned by the description level and cannot be easily adapted to other description levels as the dynamic techniques can. The probabilistic techniques that have been analyzed are restricted to gate level (§E1.2). As it has been said, the only probabilistic technique that works at RTL [38] is oriented to testability analyses; therefore, it does not estimate the switching activity. There is another RTL proposal (Wright [139]) but, although it takes the circuits at RTL, as far as we have understood, we consider that the estimation is performed at gate level after a synthesis. Therefore, **no RTL probabilistic technique for switching activity estimation has been found**.

This Thesis aims to advance the probabilistic estimation of switching activity proposing a method that estimates at RTL. Furthermore, this Thesis intends to demonstrate the advantages that RT level could contribute to the method simplification. Raising the probabilistic estimation of the activity from gate level to RTL has three implications:

- The higher the estimation abstraction level is, the earlier it can be performed. Therefore, the estimation would provide information that permits earlier decision making which could **avoid design iterations**.
- Estimations at higher abstraction levels are **simpler and faster**; nevertheless, they usually imply a **loss of exactitude** due to less information about the final circuit implementation.

- The higher a circuit abstraction level is, the more globally it can be analyzed because signals and blocks are grouped by their functionality and there is high level information that is lost during the synthesis. This fact could be used to simplify the estimation method.

The last point is fundamental in this Thesis, whose benefits will be demonstrated along the document.

Considering the RTL probabilistic estimation of the activity, the objectives of this Thesis are:

- The elaboration of a probabilistic model to estimate signal activities of RTL digital circuits
- To take advantage of RTL characteristics in order to simplify the model and thus, performing a more profitable estimation
- To propose more efficient mechanisms for calculating the switching activity

Secondary objectives of this Thesis are:

- The development of an automated tool that, taking VHDL-RTL digital circuits, elaborates the probabilistic model and estimates the activity. This tool would demonstrate the proposal validity
- To contribute to circuit quality analyses
- The integration of these estimation and quality tools in Ardid [132]; therefore, extending its functionality
- The contribution to other analyses, such as power, area, testability, ...

As it has been said, this Thesis is a first approach to RTL activity estimation by probabilistic means. Therefore, it does not intend to cover all the characteristics stated in table E1-1. The principal characteristics not being considered are gate delays, spatial correlations and sequential circuits (section E2.2).

The proposed method is presented in the next chapter. Then, the experimental results are shown in chapter E3.

E2. RTL SWITCHING ACTIVITY ESTIMATION PROPOSAL

This chapter expounds the activity estimation proposal at RTL. This work is a first approach to the probabilistic estimation at RTL; hence, it does not intend to cover all the aspects related to this abstraction level. Instead, this work attempts to demonstrate that the proposed contributions are enough to consider favorable RTL estimation against gate level estimation.

Estimating at a higher level of abstractions is advantageous not only in a methodological point of view but also because the method may be implemented in an easier way. Therefore, the benefits are both **methodological** and **practical**, and they can be summarized in:

- **Advantages related to the design methodology**: The higher the estimation abstraction level is, the sooner the final circuit characteristics can be foreseen. Therefore, designers can anticipate the fulfillment of circuit specifications and detect errors earlier, what helps in decision making that could avoid the costly iterations in the design flow.
- **Advantages related to the design simplicity**: Estimations at higher levels of abstraction are faster and simpler than those performed at lower levels. Higher level designs have less detailed information, what may simplify estimation procedures because a great part of this information is not necessary for the estimation. Nevertheless, this circuit simplicity may provoke less exactitude on estimations. But, to some extent, the circuit simplicity is due to the fact that some design decisions are not made yet and these estimation results are what may help to make those decisions.
- **Advantages related to the high level information**: Although a higher level design has less detailed information, it has other information that will not prevail after the synthesis process. This higher level information may be of great utility for the estimation procedures.

From all these advantages, it may be concluded that switching activity estimation at RTL is more favorable than at gate level. However, as it was described in last chapter, even though the previous work related to activity estimation has been very prolific, it has been restricted to gate level. None of the references we have found estimates switching activity at RTL by probabilistic means. Wright [139] claimed to estimate at behavioral level; however, although he estimated power at that level, the probabilistic method used to calculate switching activity is applied at gate level. Only Fernandes proposal [38] we have found that estimates at RTL; however, since it is oriented to testability, only signal probability is calculated but not signal activity.

This Thesis proposes to perform switching activity estimation at RTL. Hence, an estimation method for switching activity estimation is presented. This method allows demonstrating the practical advantages of RTL estimation against gate level estimation. That is, the method attempts to evidence the differences in simplicity and speed between the RTL estimation and gate level estimation. The method also tries to demonstrate the advantages of having the high level information related to RTL that is not available at gate level. The methodological advantages of estimating at RTL against gate level are demonstrated through itself, i.e. the sooner the estimation can be done (and thus the higher the abstraction level is), the better the estimator is for the design flow.

The proposal of this Thesis is wide, including both methodological aspects and particular issues that resolve specific questions. Therefore, in order to better explain the proposal, a summary is exposed in the next section (E2.1).

Since this work is a first approach, it cannot cover all the issues related to RTL. Consequently, the probabilistic model is limited. In section E2.2, all the model characteristics and simplifications are exposed.

Section E2.3 explains the circuit's structural analysis and its hardware model elaboration. This hardware model is the foundation of the probabilistic model. Since there is not a dependence graph at RTL, the hardware model is necessary for the RTL analysis. This is a significant

difference with the other approaches, because since these analyses are performed at gate level, they already have the exact circuit structure.

Later, section E2.4 explains the reconvergence analyses and the proposed partition method. Following, section E2.5 exposes how BDDs are built and the advantages of the BDDs built at RTL compared with those built at gate level. Activity BDDs are studied in detail because their size considerably increment in respect to the probability BDDs. Therefore, a new representation of activity BDDs is proposed. Then, section E2.6 shows that the RTL analysis helps to obtain better BDD orderings. Next, section E2.7 explains the method used to propagate probabilities and activities through the circuit. To end, section E2.8 elaborates the conclusions of this chapter.

E2.1 Activity estimation proposal summary

As it has been said, activity estimation methods can be classified in two categories: static and dynamic methods. Static methods, which are also known as probabilistic methods, have been chosen in this Thesis. Since no other proposal has been found that estimates switching activity at RTL, the main objective of this Thesis is to go up in switching activity estimation from gate level to RTL. Therefore, a new circuit probabilistic model for RTL activity estimation is proposed.

Estimating at RTL has the advantages that have been already commented. However, RTL estimation has also some drawbacks:

- Designs at gate level have a structure more similar to the final circuit: at gate level each signal will exist in the final circuit and the relationships among signals will also prevail. That will not necessarily happen at RTL, where signals and relationships among them may vary during the synthesis due to the circuit optimizations
- Therefore, RTL estimation needs the elaboration of a generic structure of the circuit. This structure is an approximation of the final hardware
- Since the final gate arrangement is not known at RTL, the glitch analysis is complicated at RTL

Once the proposal with its advantages and drawbacks has been defined, the estimation method will be summarized. The explanation of each part of the method will be extended in the following sections.

As it has been said, the construction of a circuit structure is required to elaborate a probabilistic circuit model. Therefore, this Thesis proposes the creation and usage of a generic structure on which the probabilistic model stands. This structure has been called *extended hardware model* (EHM). All the relationships between circuit nodes are specified in the EHM.

Once the EHM is elaborated, a structure similar to the gate level netlist is available. This structure is simpler but less accurate than the netlist but, on the other hand, it contains all the specific information related to RTL. As it will be explained, this information allows to perform analyses and simplifications that are difficult to carry out just having the gate level information. Therefore, in order to simplify the probabilistic model, this Thesis proposes to performing analysis having the RTL circuit information.

The probabilistic models of large circuits tend to excessively grow in complexity and size. To avoid it, the circuit must be partitioned. However, signal reconvergences prevent the circuit from being partitioned without losing accuracy in probability and activity computations. Hence, to maintain the accuracy, the circuit must be partitioned in reconvergent regions.

Circuit partition in reconvergent regions is a common practice for elaborating the probabilistic models at gate level. At RTL, the partition is performed in a similar way, but the partition procedure is simpler due to the less detailed information of RTL circuits. Besides, in some cases, synthesis provokes circuit reconvergences to increase what prevents the gate level circuit to be

partitioned, whilst the RTL circuit can be partitioned. Therefore, it is an advantage of RTL estimation against gate level estimation.

However, reconvergent circuit partition does not always solve the problem of complexity and size. Some circuits have so many reconvergences that they cannot be partitioned at all or the circuit partitions still remain large and complex. This is a problem for both RTL and gate level circuits; thus, other kind of partitions, which may decrease the results accuracy, must be accomplished.

In order to reduce the probabilistic model size, this Thesis proposes to divide the circuit in disjoint regions. Usually, a reconvergent region can be divided in smaller disjoint regions. Disjoint region partitions allow the calculation of exact signal probabilities and approximate signal activities; whereas other kind of partitions that have been proposed to reduce the probabilistic model size calculate both signal probabilities and activities approximately. It must be remarked that having exact values of signal probability is important because signal activities depend on both signal activity and probabilities of their dependences.

Disjoint partitions have already been used for signal probability computation at gate level [2]; however, at gate level, disjoint region identification is computationally expensive. Therefore, simulative methods have been used for their identification; nevertheless, the method cannot find all the circuit disjoint regions. This Thesis proposes to identify disjoint regions at RTL because many disjoint regions are easily identified at RTL.

The usage of disjoint partitions to compute switching activity is an original contribution of this Thesis. Since disjoint partitions do not produce exact results for signal activity, the conditions that minimize the error have been analyzed in this Thesis. This analysis helps to perform circuit partitions that minimize the error, allows the estimation of the error magnitude and, hence, evaluating whether the partition is acceptable.

Once the circuit has been partitioned, functions that calculate signal probabilities and activities can be created. For each signal, functions are built on the basis of their dependencies on other signals of the same partition. The evaluation of the function provides the values of signal probability and activity. There are different representations of probability functions, such as polynomial expressions, binary decision diagrams (BDD) or Bayesian networks (BN). BDDs have been chosen in this Thesis due to their wide usage and the availability of open source tools, which has been of importance to implement the proposed modifications.

Switching activity may be calculated applying an XOR to the probability function at two consecutive time steps (equation E1.8). The resulting function is much more complex than the original probability function and, consequently, so the activity BDDs are. Therefore, an original activity BDD (*a*BDD) representation is proposed in this Thesis. This new representation saves between 25 and 50% of the nodes of the activity BDDs. Operations to create, transform and manage the proposed *a*BDDs have been defined in this Thesis, which also are original contributions.

Another advantage of the RTL analysis is related to the BDD size. The BDD size is strongly dependent on the variable order and enormous variations in size can be found between different BDD orderings. Although several algorithms have been proposed to efficiently order BDDs, their application is computationally expensive. When gate level analysis is performed, the BDD ordering is random and, as a consequence, an unfavorable ordering may be obtained. On the contrary, near optimum orderings may be found at RTL with a minimum effort. This Thesis proposes a BDD ordering based on signal functionality extracted from RTL. This is a *natural* RTL ordering and, therefore, it does not need additional computational resources to obtain it.

In addition, for RTL operations involving arrays, such as addition, subtraction, comparison..., an optimum ordering may be predetermined depending on the array indexes. Operation and array index identification may be an immediate task at RTL; however, at gate level, it is not a straightforward task. Therefore, being able to preset the BDD ordering for the RTL operations is another RTL estimation advantage.

To end, once the EHM has been elaborated including the probability and activity BDDs, the probabilistic model is ready to calculate the signal probabilities and activities. Despite all the simplifications proposed in this Thesis, the probabilistic model may still be expensive to build for large circuits. Nevertheless, it is a one time cost, because once the model is built, it may be used for different conditions at the inputs. Thus, only numeric values have to be propagated through the probabilistic model; whilst the probabilistic model remains unchanged. This is a remarkable difference between static and dynamic methods. If input conditions change when using dynamic methods, the whole estimation process has to be repeated.

In order to facilitate the comprehension of the proposal, figure E2-1 shows a schematic representation of this Thesis proposal. The stars of the figure represent the original contributions of this work.

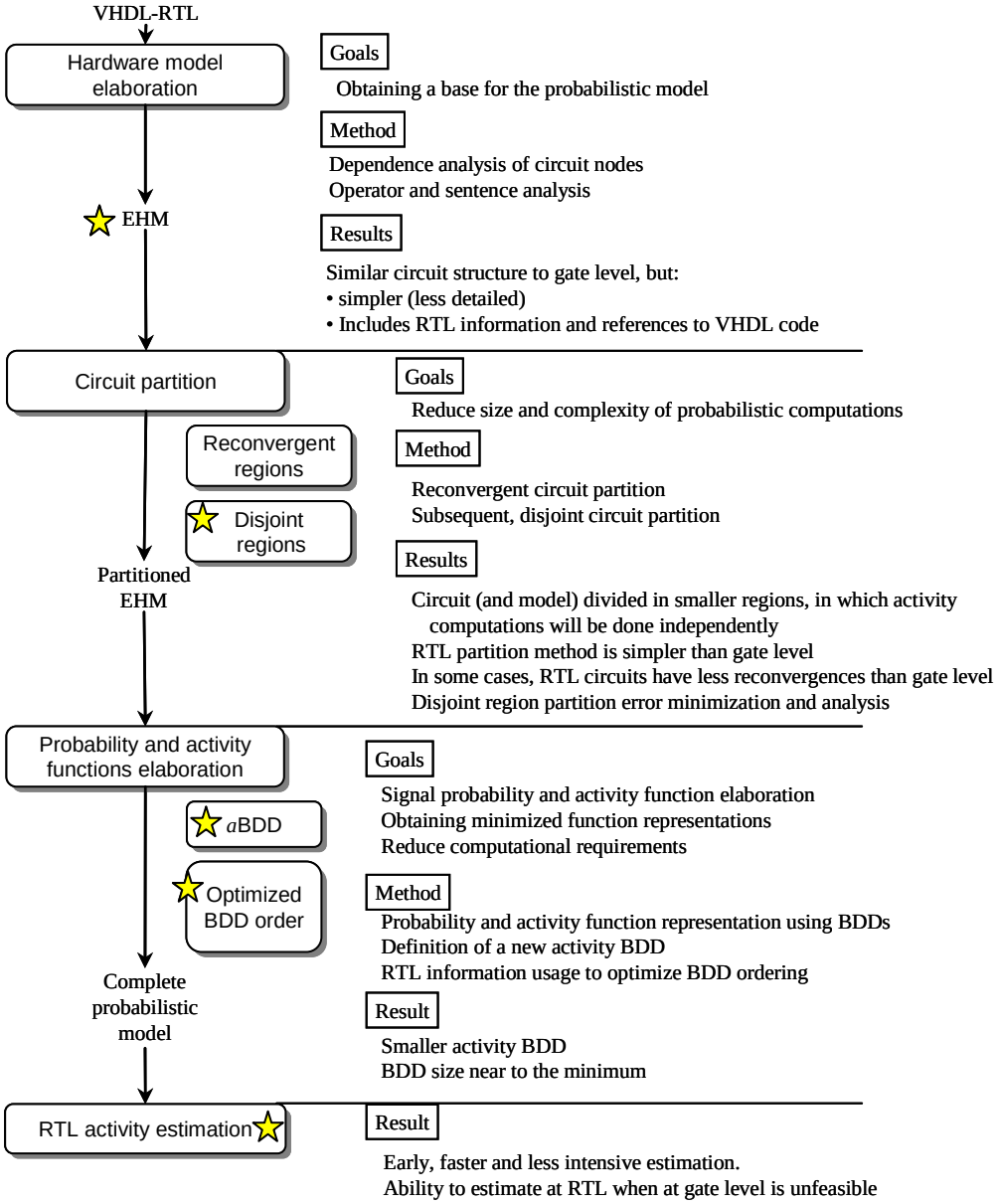


Figure E2-1: Summary of the proposal formulated in this Thesis

E2.2 Proposed model characteristics

The probabilistic activity estimation proposed in this Thesis is the first step for RTL activity estimation. Therefore, some simplifications have been adopted for the model. These simplifications do not make the model to decrease in generality. On the contrary, the inclusion of the other issues that have not been adopted may corroborate the RTL estimation advantages.

The model simplifications are:

- Only combinational circuits are considered.
- It is a zero-delay model. Therefore, glitches are not taken into account.
- A minimum time step in which no signal transition may occur is assumed. This time step is similar to the clock cycle of sequential circuits.
- Due to this minimum time step and the zero-delay assumption, signal activities can only have one transition per time step. Therefore, signal activities may be considered as a probability.
- Neither arrays nor integer number representations are considered. Only in some cases, arrays are taken into account. As a result, integers and arrays are decomposed into one bit signals, and their probabilities and activities are considered bit by bit, and not as a whole.
- An automated tool has been developed. This tool constructs the probabilistic model from VHDL designs. Since the tool is a first version, the tool just accepts a subset of the VHDL-RTL synthesis standard [60].
- Although the probabilistic model considers spatial correlations inside the circuit, spatial correlations at inputs are not taken into account.

Compared to other contributions, the main issues that are not taken into account are: delays, spatial correlations at inputs and memory elements. They have not been included to facilitate the elaboration of a simpler model. Once the benefits of RTL estimation are proved, the model could be extended.

The fact that neither all the synthesizable VHDL-RTL subset nor multibit signals have been considered does not imply a disadvantage against the other contributions because they are issues related to RTL and not to gate level.

Nevertheless, array indexes and RTL operators have been taken into account with the purpose of obtaining optimum BDD orderings.

E2.3 Circuit structural analysis and hardware model

This section explains the circuit analysis that permits the elaboration of a dependencies graph of the circuit signals. The graph will be called hardware model. This graph is the structure on which the probabilistic model stands and is the mean used for signal probability and activity propagation. The elaboration of this hardware model is necessary because at RTL there is no structure similar to gate level netlists.

The hardware model brings the RTL design closer to the gate level, providing a similar structure to the hardware that will be synthesized. Several different gate level circuits may result from RTL synthesis. This is a difficulty and limitation of RTL estimation, since the final gate level description depends on the technology library, the synthesis tool and the defined synthesis options and restrictions. For that reason, the hardware model is a generic circuit that does not intend to be as exact and detailed as the gate level circuit.

In the next subsections the steps towards the elaboration of the hardware model will be described. These steps are:

1. Simplified hardware model (SHM) construction
2. Extension of the SHM: Extended hardware model (EHM)
3. Signal combinational depth assignment

All these tasks have been automated in a computer program, what permits a higher speed in the analysis and a wider usage, especially for large circuits, for which the manual elaboration of the hardware models are unfeasible.

E2.3.1 Simplified hardware model

The *Simplified Hardware Model* (SHM [131]) is the first step towards the elaboration of the probabilistic model. This model is automatically elaborated by Ardid [132]. Ardid is a computer tool to assist digital designers developed in the *Centro de Electrónica Industrial* (CEI) at the Technical University of Madrid (UPM). The methods developed in this Thesis are integrated in Ardid.

SHM is a hardware model of VHDL designs, in which, signals and sentences VHDL are related according to their dependencies and influences. The model is constituted on three layers depending on the required level of detail:

- Sequential layer
- Combinational layer
- Statement layer

The VHDL source code related with the model elements can be accessed from any of the layers.

The **sequential layer** is the less detailed. In this layer, though the memory elements are clearly defined, the combinational logic is vaguely described. The combinational logic may be associated to a diffuse cloud in which inputs and outputs are known but not the inner logic. From each signal, the final dependences and influences can be obtained[64]. In a similar way, the final dependences of signal *X* are the ports and memory elements that influence on signal *X* through combinational logic.

Figure E2-2 shows a schematic representation of the sequential layer.

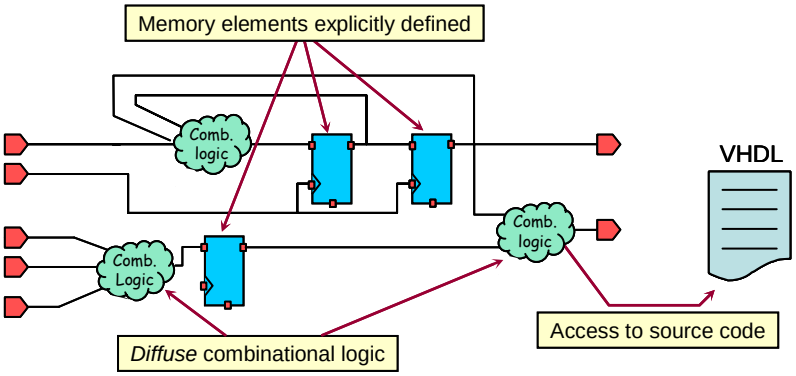


Figure E2-2: Schematic representation of the SHM sequential layer

SHM permits extracting the circuit combinational areas. These areas are demarcated by primary nodes. Ports and memory elements constitute the primary nodes of a circuit (see figure E2-3), which also compose the final dependences and influences.

[64] The final influences of any signal *X* are the ports and the memory elements that signal *X* influences through combinational logic

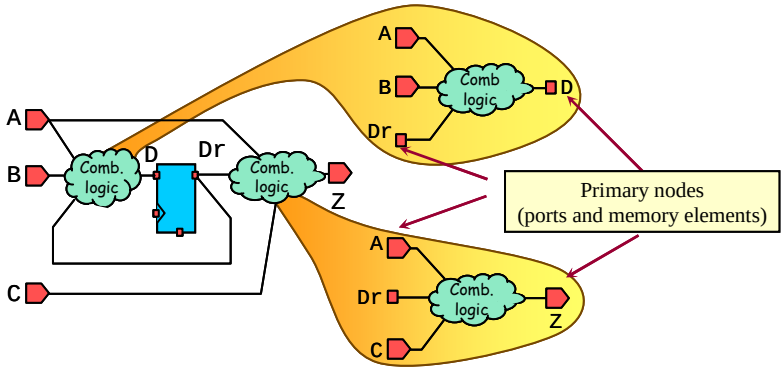


Figure E2-3: Extracting combinational areas from the sequential layer

In the **combinational layer**, those "clouds" representing the combinational logic can be analyzed. Each combinational "cloud" can be traversed through signal dependences and influences, browsing across assignments and processes. This layer only indicates signal dependencies, but does not specify the logic that interconnects them.

Figure E2-4 shows a sample VHDL combinational design and also its SHM sequential and combinational layer. Because the circuit has no memory elements, the sequential layer is just one "cloud". In the combinational layer, statements and processes are now represented by a cloud, indicating that their content is not defined in this layer. In order to facilitate the interpretation of the figure, the combinational layer has been drawn in vertical direction. Outputs (*Z*) are at the top of the figure and inputs (*A*, *B*, *C*, *D*, *E*, *F*) are at its bottom. Internal signals (*G*, *H*, *I*, *J*, *K*) are in between output and inputs (primary nodes), connected by the combinational "clouds", representing the design statements.

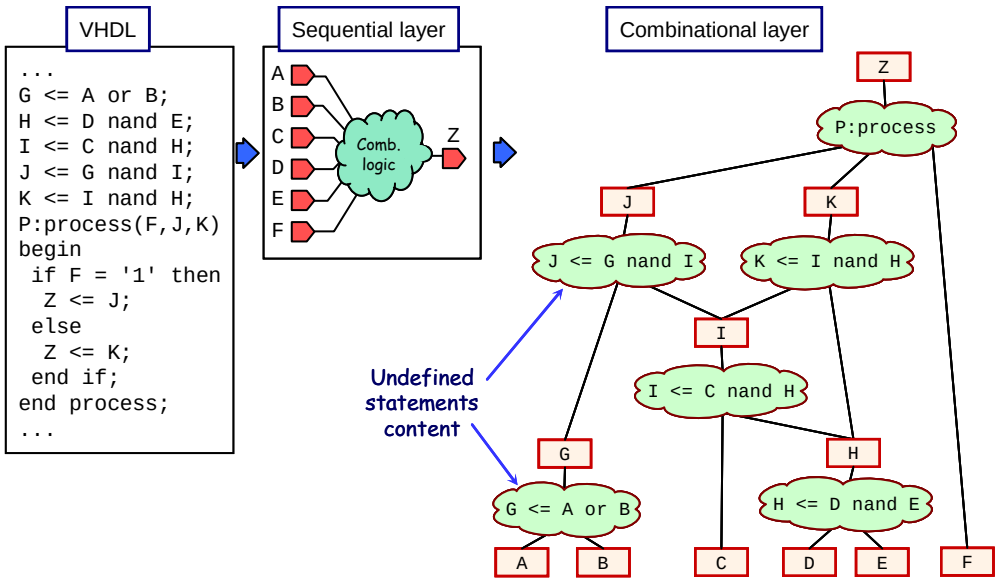


Figure E2-4: SHM sequential and combinational layers of a VHDL design

The combinational layer does not make any difference between statements and their complexity. Thus, clouds can represent either simple assign statements or complex processes. At this level, the only difference between the statements is the number of inputs and outputs. In the figure, only the signals, represented in a box, are completely defined.

The **statement layer** shows the internal expressions of statements and processes. This layer does not provide with any interpretation or simplification of the expressions. It just allows accessing the statements in the same way they were described in VHDL.

The statement layer of one of the statements of figure E2-4 has been represented in figure E2-5. The differences between the combinational and the statement layer can be appreciated in this figure.

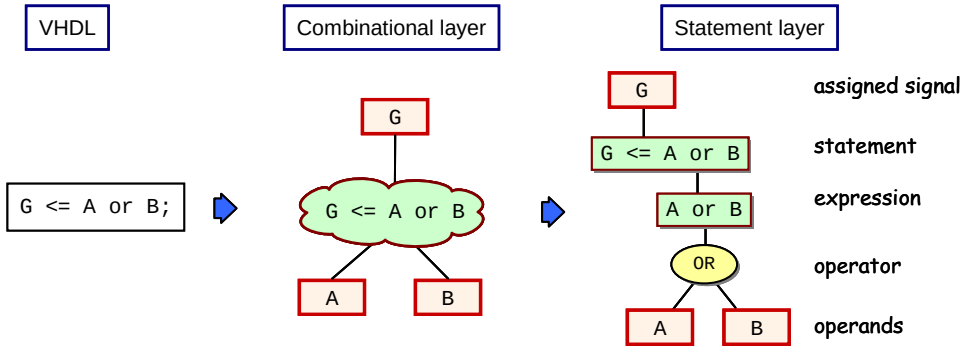


Figure E2-5: Combinational and statement layers for a simple assign statement

Processes have more complex statement layers because signals may be assigned in more than one statement and they may be influenced by various conditions. Figure E2-6 shows the combinational and statement layers of the process of figure E2-4. In this image, dashed lines join statements with their conditions. If conditions are negated (i.e. else statement) the dashed line ends in a black dot.

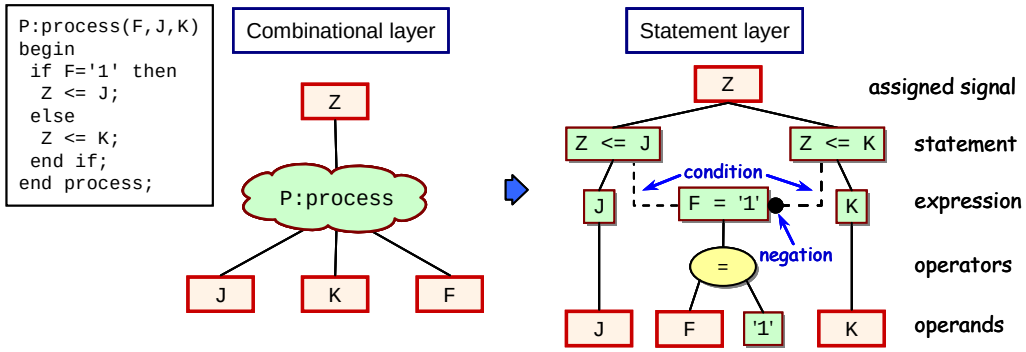


Figure E2-6: Combinational and statement layers of the process of figure E2-4

The previous example is elemental. Figure E2-7 shows a more complex example, in which signal *Z* is assigned in three statements. Therefore, in the figure, these three statements lead from signal *Z*. These statements are affected by the conditional expressions.

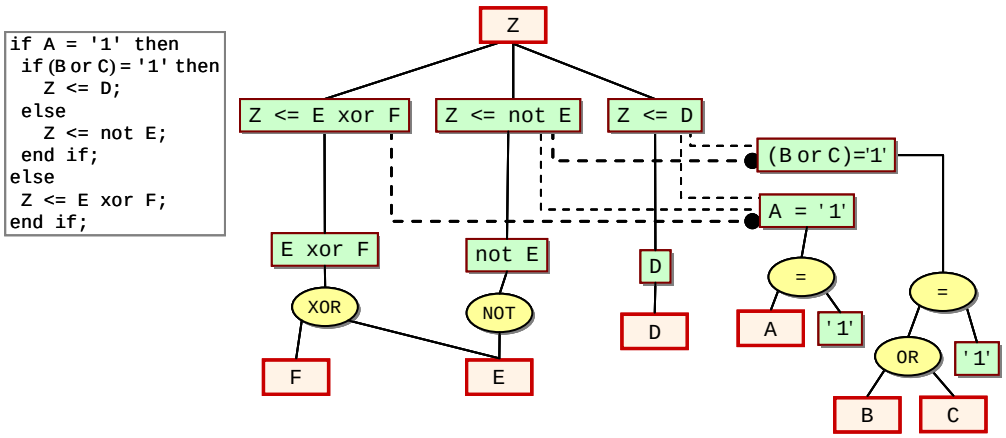


Figure E2-7: Statement layer of a sample process

E2.3.2 Extended hardware model

The simplified hardware model (SHM) is very useful because signal influences and dependences are easily reached. The usage of the sequential layer leads to a signal's final influences and dependences. The combinational layer leads to a signal's immediate influences and dependences.

However, the information given by the statement layer is not easy to use. Particularly, conditional expressions are not effectively processed. For example, the statement layer of the process of figure E2-7 is too complicated to extract the hierarchy of the conditions and assignments.

Because conditional statements are very common in VHDL, it is necessary to extend the SHM to enhance its representation. As a result, an *extended hardware model* (EHM) has been developed in this Thesis.

The proposed extension is shown in an example in figure E2-8. To facilitate the comparison between both models, the same process of figure E2-6 has been analyzed in figure E2-8.

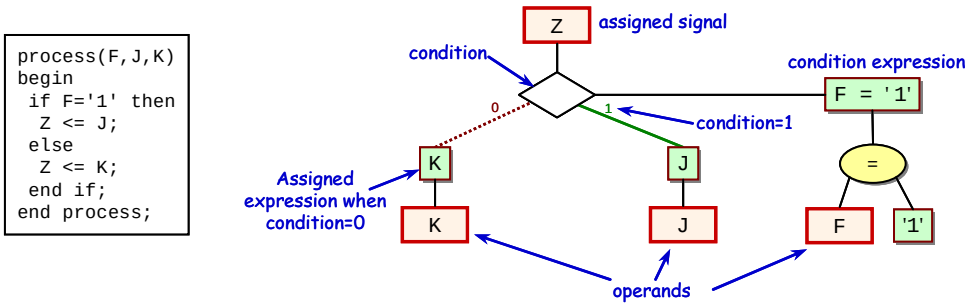


Figure E2-8: Process representation in EHM (same process as in figure E2-6)

In the EHM, only the conditional statement (if *F*='1' then) hangs from signal *Z*. Three edges hang from the condition, one of them horizontal, the others sloped:

- The horizontal edge leads to the condition expression (*F*='1')
- The sloped, doted edge leads to the expression that will be assigned in case the condition is not satisfied.
- The sloped, solid edge leads to the expression that will be assigned in case the condition is satisfied.

The EHM of a more complex process is represented in figure E2-9. The process is the same process that has been represented in figure E2-7. The process has two conditional statements. The outer conditional statement leads from signal *Z*.

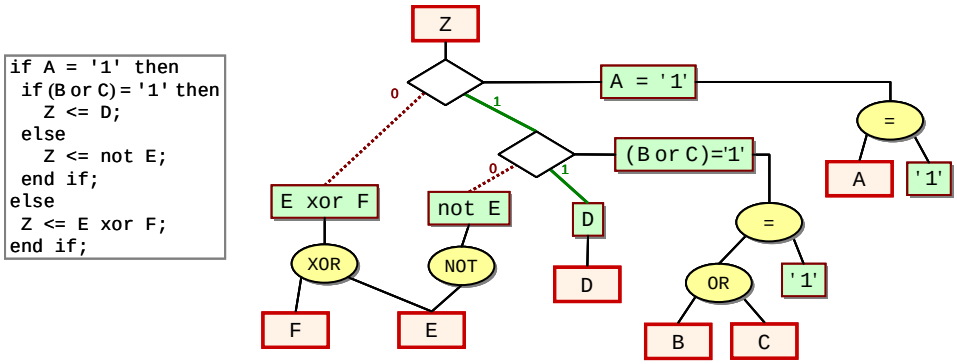


Figure E2-9: EHM of the process of figure E2-7

From the EHM is easy to extract the hierarchy of statements and conditions. The EHM is more similar to the hardware that will be synthesized. As it can be seen from figure E2-10, EHM conditions represent multiplexers in hardware.

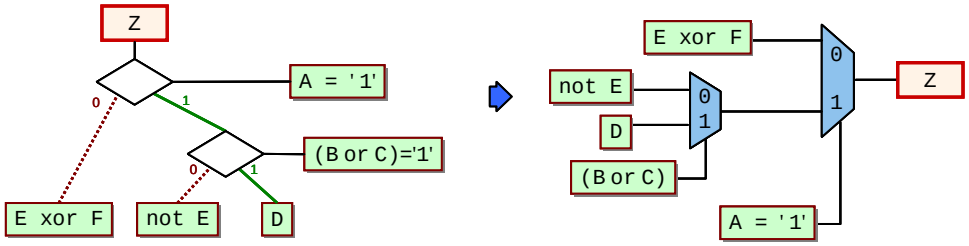


Figure E2-10: EHM to hardware equivalence

Consequently, the EHM is an approximation to the hardware that will be synthesized and provides an efficient structure for the activity computations.

E2.3.3 Combinational depth

Once the extended hardware model is ready, a signal hierarchy can be established depending on their influence and dependence relationships. Following this hierarchical order among signals, it can be ensured that before analyzing any signal, all its dependences have been already analyzed.

In order to establish this hierarchy, every signal is assigned a *Combinational Depth* (*CD*) indicating the distance to its farthest final dependence.

Final dependences (primary nodes) have zero combinational depth (CD=0). Every other signal has a combinational depth higher than zero. The combinational depth (CD_X) of any signal *X* is the highest combinational depth of all its immediate dependences plus one.

$$CD_X = MAX_{i \in Inm_X} \{CD_i\} + 1 \qquad \text{(E2.1)}$$

Where Inm_X is the set of all the immediate dependences of signal *X*.

As an example, figure E2-11 shows the circuit signals combinational depth of the circuit already drawn in figure E2-4

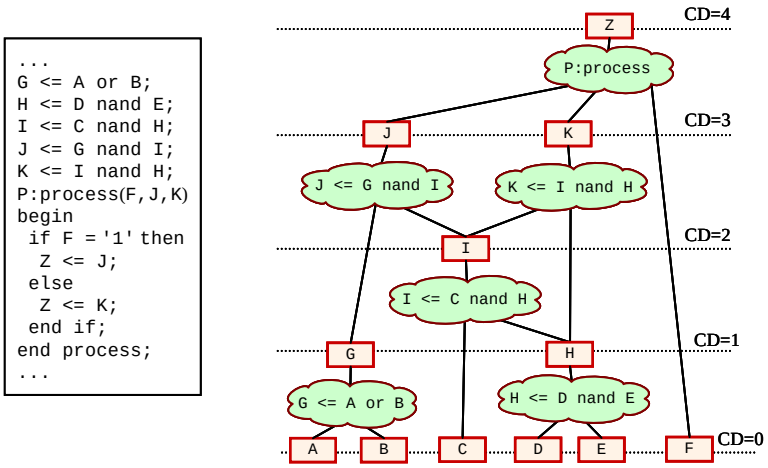


Figure E2-11: Combinational depth of the circuit of figure E2-4

At RT level, the circuit combinational depth would be represented as shown in figure E2-12

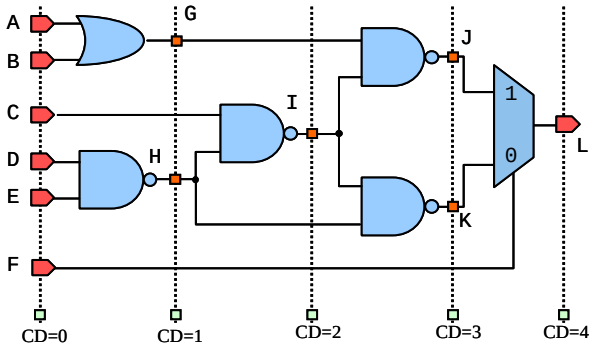


Figure E2-12: Combinational depth for the circuit of figure E2-4 represented at RTL

During the analysis, signals with lower combinational depth would be first analyzed. Consequently, signals *G* and *H* would be analyzed before than signal *I*. After signal *I*, signals *J* and *K* would be analyzed. To end, signal *L* would be the last signal to be analyzed. This kind of organization has been previously used by other authors for power estimation, which has been called *levelization* [98], [23].

E2.4 Circuit partition

Once the EHM is elaborated, a structure to build probability and activity functions is available. Signal probability and activity can be now propagated by means of these functions through the EHM from inputs to outputs.

However, the probabilistic models tend to grow enormously with circuit size. Therefore, circuit partitioning is necessary to limit the usage of memory and computational resources. Nevertheless, in order to build an exact probabilistic model, the circuit should be partitioned considering structural dependences (figure E1-1).

Structural dependences analysis leads to *reconvergent regions partition* (or just *reconvergent partition*), which was explained in figure E1-5 (also called *supergates* [121]). In large circuits, reconvergent partitions may still result excessively large, and hence, the reconvergent regions must be limited (figure E1-6).

As a consequence, in addition to the reconvergent partition, this Thesis proposes to divide the circuit in disjoint regions. The reason is that disjoint region partitions (or just *disjoint partitions*) divide reconvergent partitions in smaller regions.

The next sections will explain:

- Disjoint regions identification at RTL
- Switching activity calculation in disjoint regions
- Error analysis due to disjoint partitions

E2.4.1 Disjoint regions identification

Most of the proposals look for independent signals for circuit partitioning; in consequence, they perform reconvergent partitions. Only Agrawal proposes [2] to look for disjoint signals to divide circuits in smaller disjoint regions. Since his proposal is restricted to gate level, the method to find disjoint signals is not straightforward.

In contrast, the identification of disjoint signals may be effortless at RTL. RTL conditional statements may create disjoint signals and, as a result, they may be immediately detected. Generally, conditional statements represent multiplexers in which the condition selects what

signal is assigned. Multiplexers generate mutually disjoint signals. Figure E2-13 shows the different representations of a multiplexer: in a hardware description language (VHDL), at RTL, in the extended hardware model and at gate level.

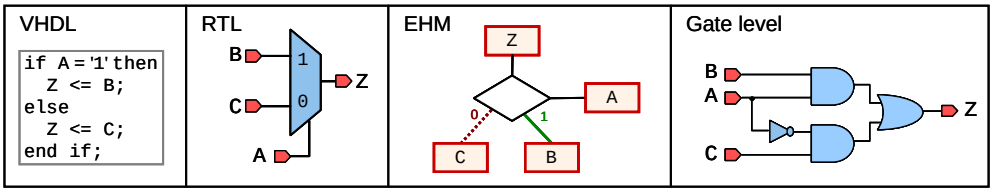


Figure E2-13: Different representations of a multiplexer

Identification of multiplexers is straightforward at RTL or with the EHM, as it can be seen from figure E2-13. On the contrary, at gate level is not easy because gates may have a different arrangement from the one of figure E2-13, especially in larger multiplexers.

E2.4.1.1 Disjoint signals in multiplexers

By definition, multiplexers produce nodes whose events can never happen simultaneously. A multiplexer selects one and only one of the inputs to assign it to the output. Therefore, mutually disjoint signals are generated inside multiplexers.

Similarly to multiplexers, conditional sentences also produce mutually disjoint events. Alternatives of "if statements" are exclusive because only one can be selected at a time. Figure E2-14 shows the disjoint events (*D* and *E*) of a simple conditional statement that represents a multiplexer. Events *D* and *E* cannot happen simultaneously, because it cannot happen that *A*='1' and *A*='0' at the same time.

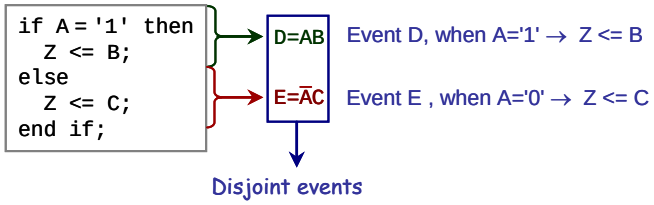


Figure E2-14: Disjoint events in a conditional statement (multiplexer)

Since nodes *D* and *E* are disjoint, the probability function of signal *Z* can be expressed in terms of *D* and *E*. This step does not imply any error (figure E2-15):

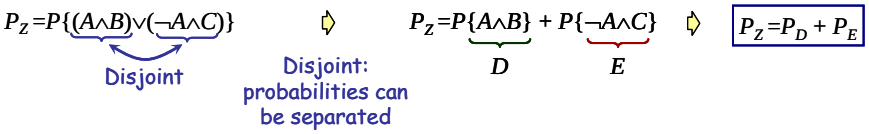


Figure E2-15: Separation of probabilities of disjoint events

Figure E2-16 shows where nodes *D* and *E* are located in the circuit at gate level.

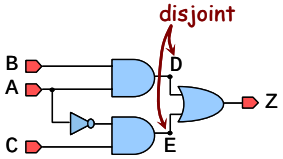


Figure E2-16: Multiplexer's disjoint nodes at gate level

It is important to say that this calculation can be carried out even when input signals are not independent. This calculation is not approximate, that is, the resultant probability values are exactly the same as the ones obtained from reconvergent partitions.

In digital circuits, it may be frequent to have datapaths independent of control signals. In these cases, output signal probability can be expressed in function of input signals, instead of having it expressed in function of internal nodes that do not exist in the VHDL code or the RTL representation (as nodes *D* and *E* would be).

Figure E2-17 shows an example of this case. Multiplexer alternatives (*B* and *C*) do not have to be mutually independent, but they are independent of condition *A*.

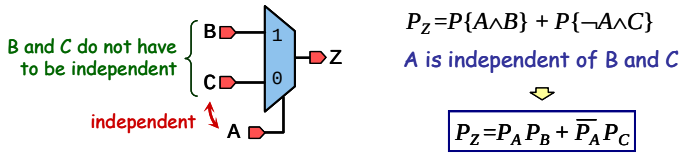


Figure E2-17: Probability calculation with an independent condition

E2.4.1.2 Disjoint partition examples

Some simple examples of disjoint partitions will be shown in this subsection.

Figure E2-18 shows an example where reconvergent signal *C* affects to both multiplexer alternatives (*I* and *H*). Therefore, if the circuit were partitioned in reconvergent regions, independent signals would be searched. As a result, the reconvergent region would embrace the whole circuit (see central circuit of figure E2-18). On the other hand, the disjoint region only contains the multiplexer (right circuit of figure E2-18). Only Agrawal [2], at gate level, would take the disjoint region if his algorithm did detect it (because the algorithm does not identify all the disjoint signals). On the contrary, identifying this kind of disjoint regions is straightforward at RTL.

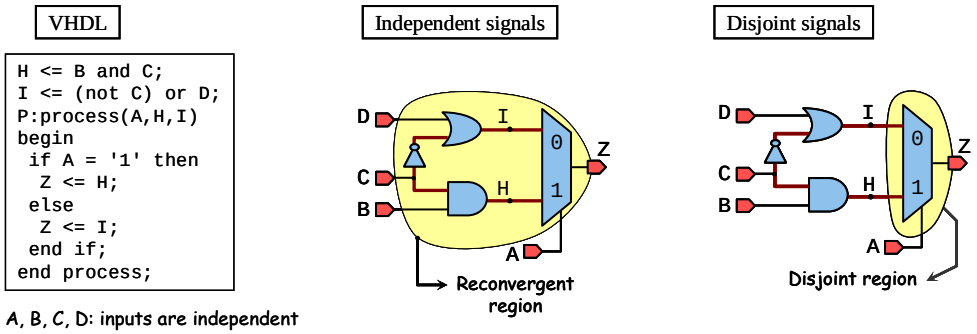


Figure E2-18: Region partition for signal Z considering independent signals (center) or disjoint signals (right)

Figure E2-19 shows another example of reconvergent partition and disjoint partition. The reconvergent partition is larger because signals *J* and *K* are not independent because they depend on two common signals: *G* and *H*.

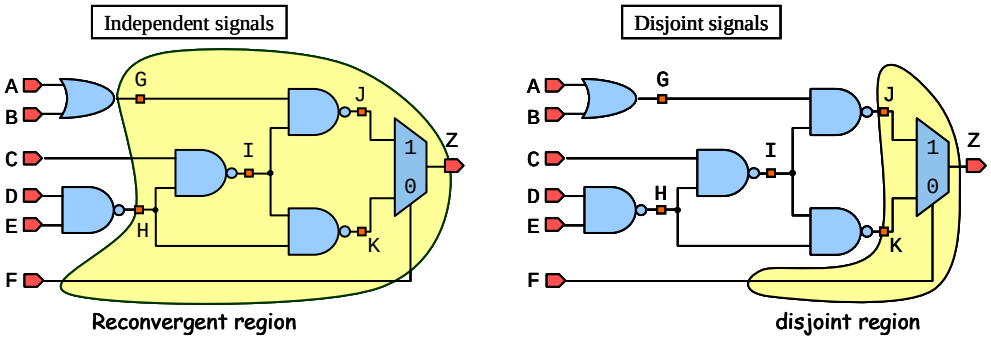


Figure E2-19: Reconvergent region and disjoint region of the circuit of figure E2-4

In figure E2-20, a last example shows that the partition can be done even inside if statements. In order to facilitate the comprehension, internal nodes (*K*, *L* and *M*) have been marked in the VHDL code. Nested if statements can be substituted by internal nodes, as internal node *K* does representing the inner if statement.

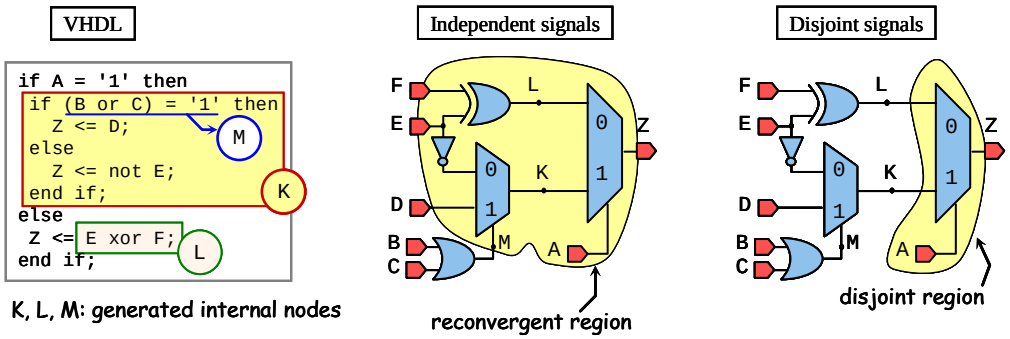


Figure E2-20: Reconvergent region and disjoint region in a process with nested if statements

Nested conditional statements do not complicate the method. They are equivalent to the decomposition of the nested statement in concurrent statements. From the previous example, figure E2-21 shows how this decomposition may be carried out. Signals *K* and *L* have been created to represent the internal nodes. Signal *K* assignment is accomplished in another process (*inner process*) and signal *L* is assigned in a concurrent statement.

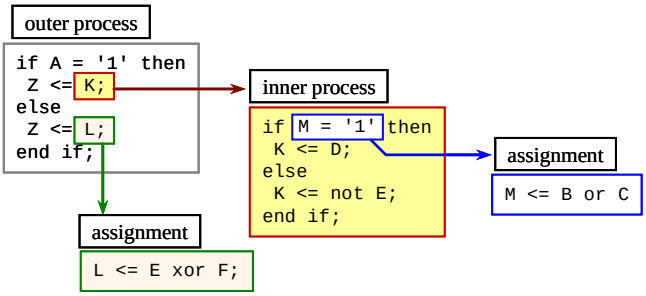


Figure E2-21: Process decomposition in concurrent statement

E2.4.2 Activity calculation in disjoint regions

As it has been explained, disjoint partitioning is an effective practice to simplify signal probability computations. This method does not provoke any accuracy loss for signal probability computation; however, some errors are produced in activity computations when disjoint partitions are used.

The activity equation for a simple multiplexer such as the one in figure E2-17 can be built applying the *Shannon expansion* [119] to the activity function and cofactoring the function with the selection signal *A*.

$$a_Z = P(A_{1\to1})\cdot a(Z)_{A_{1\to1}} + P(A_{0\to0})\cdot a(Z)_{A_{0\to0}} + \\ + P(A_{0\to1})\cdot a(Z)_{A_{0\to1}} + P(A_{1\to0})\cdot a(Z)_{A_{1\to0}} \qquad \text{(E2.2)}$$

If disjoint partition has been carried out, cofactors $a(Z)_{A_{0\to1}}$ and $a(Z)_{A_{1\to0}}$ depend exclusively on alternatives *B* and *C*, but not on selection signal *A*. However, when signals *B* and *C* are not independent, to avoid provoking error, cofactor calculation should be done in function of their independent sources. Therefore, even that *B* and *C* are disjoint signals, when they are dependent signals they produce inexact switching activity calculations.

An example will illustrate the effect of disjoint and dependent signals on the switching activity error. In figure E2-22, multiplexer alternatives *B* and *C* have a common source *D*. Therefore, *B* and *C* are no mutually independent, but they are independent to condition *A*.

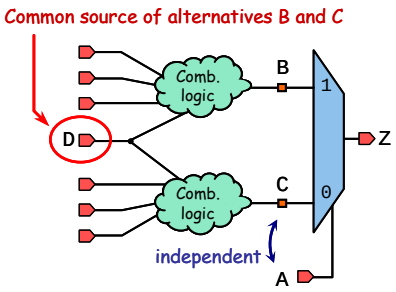


Figure E2-22: Multiplexer alternatives with a common source (D)

When the selection signal *A* remains unchanged during two time steps (*t*=0, *t*=*T*), output *Z* receives the value from the same alternative (*B* or *C*) during both time steps. Therefore, the activity of signal *Z* will be the same as the activity of the selected alternative. This case corresponds to cofactors $a(Z)_{A_{0\to0}}$ and $a(Z)_{A_{1\to1}}$

Figure E2-23 shows the case when signal *A* remains at zero during two consecutive time steps. In this condition, the circuit is equivalent to the circuit drawn on the right, where signal *Z* directly receives the value from signal *C*. Therefore, the common source *D* does not affect to the calculation. This case corresponds to cofactor $a(Z)_{A_{0\to0}}$.

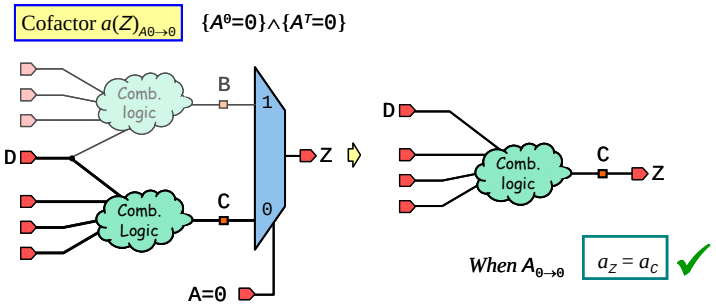


Figure E2-23: Activity of Z when the selection signal remains at zero: cofactor $a(Z)_{A_{0\to0}}$

Similarly to the previous situation, when signal *A* remains at '1' during two consecutive time steps, the activity of *Z* will be the same as the activity of alternative *B*. This case corresponds to cofactor $a(Z)_{A_{1\to1}}$.

When selection signal *A* suffers a transition from one time step to another, output *Z* will receive the value of the different alternatives during those time steps. If alternatives are independent, the value of one alternative has no influence on the other alternative at the next time step. Nevertheless, if they have a common dependence, this dependence may influence on the other alternative due to the temporal correlations.

Next, the reasons why activity calculation errors are produced when alternatives are not independent will be explained (see figure E2-24)

- Signal *D* influences on both alternatives *B* and *C*
- Due to temporal correlations, signal *D* at time *T* (D^T) depends on its value at time 0 (D^0)
- If selection signal *A* experiments a transition from 0 to 1 ($A_{0\to1}$), then
  - At time *t*=0 alternative C^0 is assigned to Z^0
  - At time *t*=*T* alternative B^T is assigned to Z^T
- Therefore, the activity of signal *Z* (that is, a_Z) depends on alternatives C^0 and B^T
- As a consequence, it is not exact to calculate the activity of *Z* from the probability values of alternatives C^0 and B^T because they have a common dependence: D^0
  - C^0 depends on D^0
  - B^T depends on D^T, and D^T is conditioned on D^0

As a conclusion, making a disjoint partition in multiplexers when the alternatives are not independent may produce approximate activity results. The reason is that the partition may provoke inaccuracies in cofactors $a(Z)_{A_{0\to1}}$ and $a(Z)_{A_{1\to0}}$.

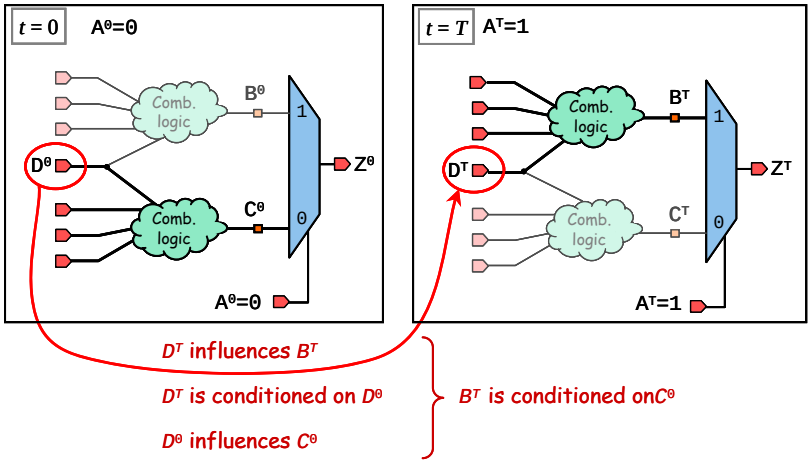


Figure E2-24: How a common dependence (D) provokes an activity calculation error on the output signal (Z)

Now, the resultant equations from using disjoint regions will be described.

The cofactor $a(Z)_{A_{0\to1}}$ equation of the circuit of figure E2-24 is:

$$a(Z)_{A_{0\to1}} = P\{(C^0=1)\wedge(B^T=0)\} + P\{(C^0=0)\wedge(B^T=1)\} \quad \text{(E2.3)}$$

If common dependence D is not considered, probabilities can be decomposed. The resultant approximate equation is:

$$a(Z)_{A_{0\to1}} \approx P\{C^0=1\}\cdot P\{B^T=0\} + P\{C^0=0\}\cdot P\{B^T=1\} \quad \text{(E2.4)}$$

Since the probabilistic model has been considered as strict-sense stationary (equation E1.1):

$$a(Z)_{A_{0\to1}} \approx a'(Z)_{A_{0\to1}} = P_C\cdot\overline{P_B} + \overline{P_C}\cdot P_B \quad \text{(E2.5)}$$

When it is necessary, an apostrophe (') will be added to the approximate activities in order to difference them with the exact activities. Thus, a'_Z or $a'(Z)$ represent the approximate activity of signal Z. The activity is approximate due to the usage of disjoint partitions when alternatives are not independent.

In order to summarize cofactors equations and approximations, figure E2-25 shows the four cofactors in respect to A: $a(Z)_{A_{i\to j}}$. The upper two cofactors are related with the inactivity of A. Therefore, they have been called inactivity cofactors. The other two cofactors are related with the activity of A and then, they have been named activity cofactors.

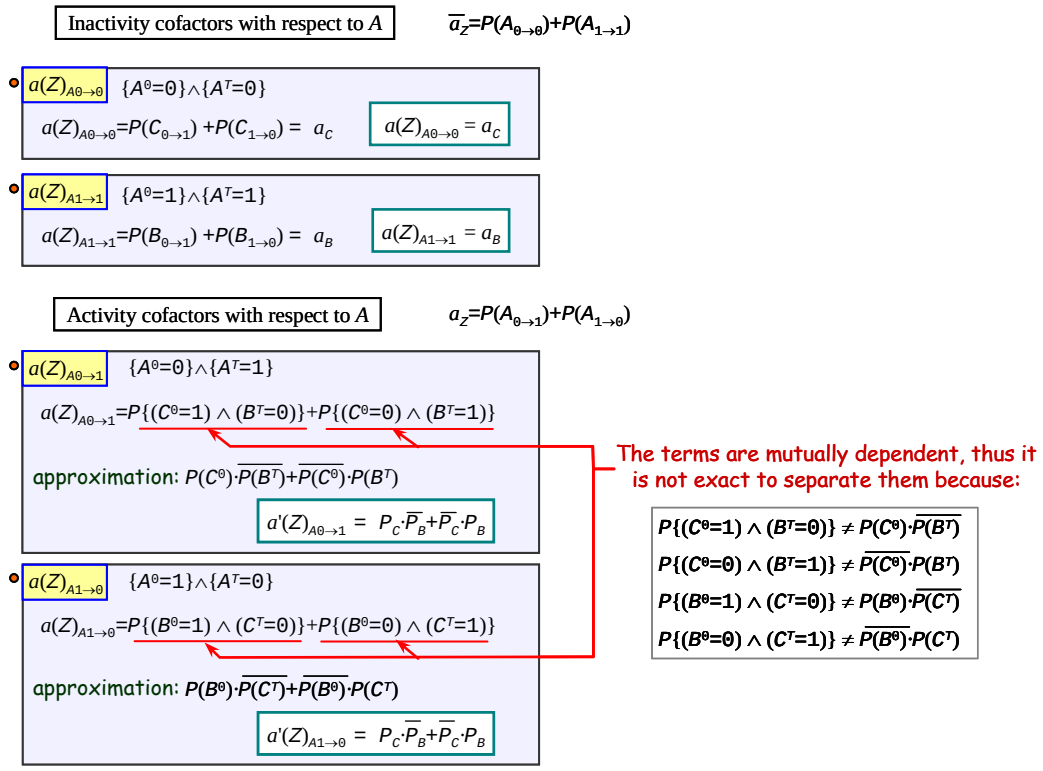


Figure E2-25: $a(Z)_{A_{i\to j}}$, the four cofactors in respect to selection signal A of the multiplexer of figure E2-22

Taking equation E2.2 and those in figure E2-25, the approximate activity equation is:

$$a_Z \approx a'_Z = P(A_{1\to1})\cdot a_B + P(A_{0\to0})\cdot a_C +$$
$$+ P(A_{0\to1})\cdot [P_C\cdot\overline{P}_B + \overline{P}_C\cdot P_B] + P(A_{1\to0})\cdot [P_C\cdot\overline{P}_B + \overline{P}_C\cdot P_B]$$

Using equation E1.3, it can be transformed into:

$$a_Z \approx a'_Z = P(A_{1\to1})\cdot a_B + P(A_{0\to0})\cdot a_C + a_A\cdot [P_C\cdot\overline{P}_B + \overline{P}_C\cdot P_B] \quad \text{(E2.6)}$$

The first two terms of equation E2.6 are exact and the last term is approximate. It should be stressed the simplicity of the resulting formula, which only depends on the multiplexer inputs. Therefore, it is not necessary to look for independent signals and the usage of large reconvergent regions is avoided. Next section analyses the error magnitude caused by the usage of disjoint regions.

E2.4.3 Analysis of the activity error from using disjoint regions

From equation E2.6 it can be deduced that the smaller the selection signal activity is, the smaller the influence of the approximation will be. The reason for this, is that the last term of equation E2.6 is the approximate one and it is the one related with the selection signal activity.

Besides, last section showed that the activity calculation is approximated when reconvergences affect the inputs of the multiplexer.

Therefore, in order to minimize the error, some conditions have been deduced:

1. The selection signal should be independent of the alternatives
2. The selection signal should have low activity
3. Reconvergent signals should have low temporal correlations
4. There should be a low rate of common sources for the alternatives

These four conditions are summarized in figure E2-26.

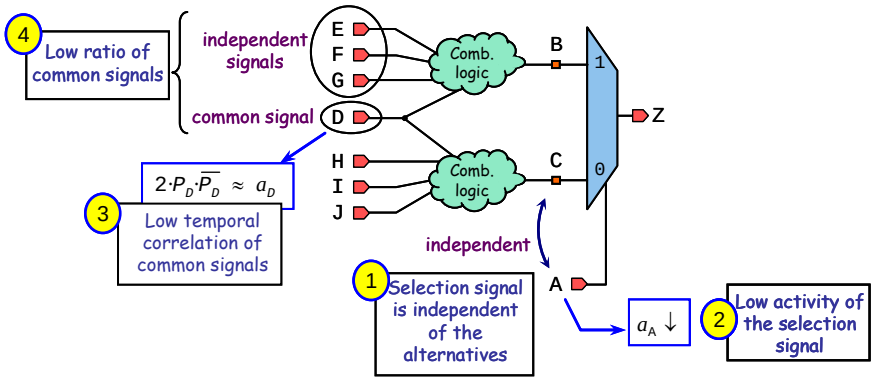


Figure E2-26: Conditions that minimize the activity error when circuits are partioned in disjoint regions

Now the fundamentals of these conditions will be explained.

It is not necessary to fulfil all the conditions but the first condition is considered indispensable. That is, the selection signal has to be independent of the alternatives.

When the **first condition** is met, only the activity cofactors (equation E2.5 and figure E2-25) are approximate. Therefore, only the last term of equation E2.6 is approximate. The fulfillment of this condition justifies the second condition.

The **second condition** imposes a low activity of the selection signal. The lower the selection signal activity is, the less the activity cofactors influence on the activity equation (equations E2.2 and E2.6). Due to the first condition only the activity cofactors are approximate; therefore, the error is reduced because this condition minimizes the influence of the activity cofactors.

When **third condition** is met, the error drops due to the low temporal correlations of common signals. If common signals have no temporal correlations (equation E1.5) there will be no error. This is because the simplifications carried out to obtain equation E2.6 are based on this approximation: the activity of the common signal of the activity cofactor is considered to have no temporal correlation.

The **fourth condition** indicates that should be a low rate of common signals influencing the alternatives. Therefore, the influence of the reconvergences would be low. This is because only the activities of the common signals are approximated.

These four conditions can be classified in **structural conditions** and **extrinsic conditions**. The first and the fourth conditions are structural because the fulfillment of these conditions depends on the circuit structure; therefore, they are invariant for a specific circuit. Conditions second and third depend on the probability and activity values of the signals. Thus, conditions may vary with the external circumstances; hence, they have been denominated extrinsic conditions.

The fulfillment of these conditions must be analyzed when a circuit has to be partitioned. During the circuit partition process, the necessary information to evaluate both structural and extrinsic conditions can be obtained by means of the EHM (§E2.3.2). In the experimental results chapter it will be shown that circuits can be partitioned in very small disjoint areas. Nevertheless, due to the error that disjoint partitions produce, these partitions should not be applied exhaustively. On the contrary, circuits must be partitioned only in those nodes whereon the error is minimized due to the circuit structure and conditions. A few strategic disjoint partitions may extraordinarily reduce BDD sizes maintaining the error low.

In consequence, **exhaustive disjoint partitions** should be avoided. Exhaustive disjoint partitions are those resulting from partitioning the circuit in the maximum number of disjoint partitions; hence the resulting partitions will be small. The experimental results chapter will show that this partitioning strategy is not convenient because the error rises and because having too many small regions do not produce any practical advantage.

On the contrary, a strategy based on **minimum disjoint partitions** is the recommended one. Minimum disjoint partitions are those circuit partitions resulting from applying disjoint partitions exclusively in the nodes where:

- The resulting disjoint regions are significantly smaller than the original
- The resulting disjoint regions remain large but manageable
- The analysis of the error conditions for the disjoint partition is favorable

In the experiment results chapter there are some examples of the effect of both exhaustive and minimum disjoint partitions.

E2.4.4 Conclusions on circuit partitioning

In addition to the reconvergent partition, this Thesis proposes circuit partitions using disjoint regions. Disjoint regions coming from conditional statements of hardware description languages offer an efficient and exact means for signal probability calculations. Disjoint regions are smaller than reconvergent regions. For example, in multiplexers, disjoint regions are limited to the multiplexer alternatives. On the other hand, reconvergent regions of multiplexers have to reach the independent signals. These reconvergent regions could reach the circuit inputs, provoking an excessive need of computational resources. As it will be shown in chapter E3, disjoint partitions may achieve important reductions in area and BDD sizes.

Although disjoint partitions may produce important errors on **activity** computations, the error can be assumed when some of the proposed conditions for partitioning are fulfilled and a **minimum disjoint partition** strategy has been followed.

E2.5 BDD construction

Previous sections have explained how the circuit model is built (EHM §E2.3) and then, how the circuit is partitioned (§E2.4). Therefore, a convenient circuit model is available, which has been divided to reduce the required computational and memory resources.

The probability and activity functions of the circuit signals are obtained by means of binary decision diagrams (BDD). Other methods apart from BDDs exist, for example, polynomial representations [103] or Bayesian networks [6]. However, BDDs have been chosen by their widespread usage and the existence of open source software to create and manipulate BDDs.

Prior to build the probability and activity functions of any signal, the probability and activity values of its sources has to be calculated.

The objective of this section is to explain how obtain the activity functions and how activity BDDs are built.

Since temporal correlations are considered, activity computations are much complex than probability computations. That is the reason why both activity functions and BDDs significantly increment in complexity and size.

For this reason, **this Thesis proposes a new manner to represent activity functions in a BDD**. This proposal reduces activity BDD sizes from 25% to 50%. Before explaining the proposal, in the next subsection the common way to represent activity BDDs is explained. Then, subsection E2.5.2 exposes the proposed method. Last, in subsections E2.5.3 and E2.5.4 compares both alternatives.

E2.5.1 Activity BDDs based on time (TFBDD)

This section explains activity BDDs based on time, which originally were named *Transition-function BDD* (TFBDD) and were proposed by Schneider [119]. These activity BDDs result from applying an XOR to the probability function at two consecutive time steps (equation E1.8). This operation extraordinarily increments function and BDD sizes. TFBDDs are explained in order to compare them with the activity BDDs proposed in this Thesis.

Next, TFBDDs structure and interpretation will be explained (§E2.5.1.1). Following, section E2.5.1.2 will expose how to obtain the activity function from a TFBDD.

E2.5.1.1 TFBDD structure

As it has already been introduced in section E1.2, the BDD resulting from applying the XOR to the probability BDD at two consecutive time steps is a BDD based on time: the TFBDD. TFBDDs are ordered maintaining the original order, but each variable of any signal at time 0 is followed by the variable corresponding to the same signal at the next time step (t=T). The new variables corresponding to time T have to be created. In TFBDDs, nodes corresponding to the same signal but at different time have to be evaluated jointly.

Before continuing, a brief explanation will be given to differentiate the terms signal, node and variable. Figure E2-27 shows a graphic representation of these three terms (it is the same TFBDD as the one of figure E1-7)

- **Signals** are the outputs and inputs of the logical and structural elements in a circuit. They are often known as circuit nodes; however, this term will be avoided in this Thesis in order to differentiate them from BDD nodes.
- **Nodes** are the BDD circles, which are also known as vertices. BDDs always have two terminal nodes, which are represented by a square. When the number of BDD nodes is counted, in this Thesis the terminal nodes are not considered.
- **Variables** are the different labels of the BDD nodes. Note that, for BDDs based on time, more than one variable may exist for each circuit signal. For example, the TFBDD in figure E2-27 has two variables (A^0 and A^T) for signal A.

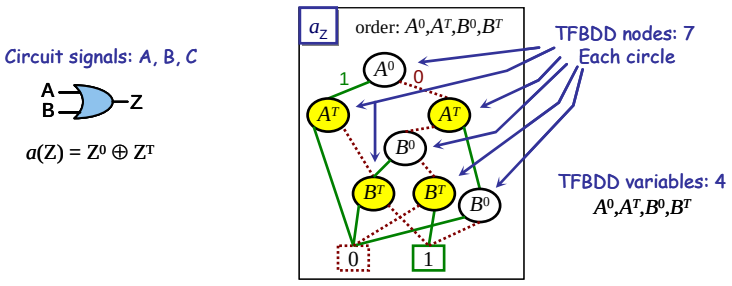


Figure E2-27: Differences between circuit signals, BDD nodes and variables

The general TFBDD structure for any signal A is shown in figure E2-28. The part of the TFBDD concerning signal A is inside a rectangle. On top of node A^0 there is any node x^t related to other signal at any time (0 or T). Similarly, at the bottom of nodes A^T there is any other node, which may be different for each edge. What is important now is the rectangle's content. Traversing the rectangles nodes from node A^0, the four transition possibilities of signal A are obtained: $A_{0\to0}$, $A_{1\to1}$, $A_{0\to1}$, $A_{1\to0}$.

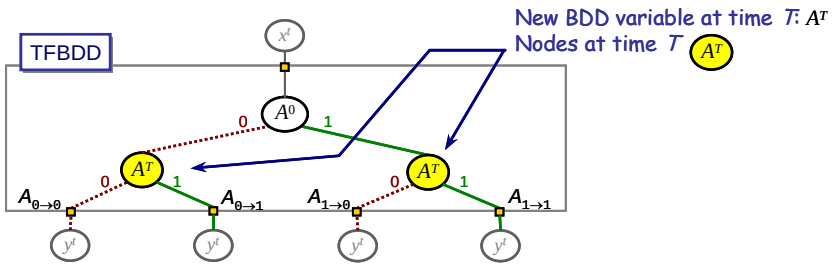


Figure E2-28: TFBDD general structure for signal A

These four transitions are obtained by the joint evaluation of each of the four possible paths that starts from node A^0 and end in the rectangle's four bottom edges (figure E2-29). For example, taking edge zero of node A^0 means that $A(t=0) = 0$. The edge one of node A^T means that $A(t=T) = 1$. Therefore, traversing the diagram from A^0 through these two edges leads to $A_{0\to1} = \{(A^0 = 0) \wedge (A^T = 1)\}$.

On the right of figure E2-29, the four paths that can be traversed have been drawn separately.

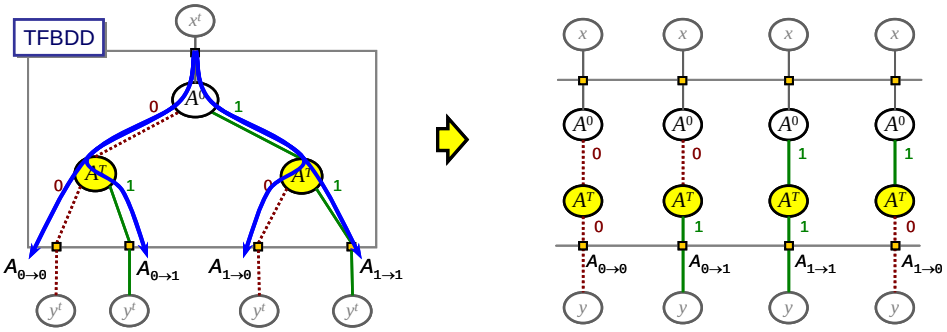


Figure E2-29: The four paths in a TFBDD general structure for signal A

Figure E2-28 shows the TFBDD general structure for a signal. Nevertheless, sometimes, simplified structures, resulting from the BDD simplifications rules, may appear. These reductions decrease the number of BDD nodes and paths; thus, BDD sizes also decrease.

Figure E2-30 shows a reduction example. In the left structure, both edges of the rightmost node A^T end in the same node. As a consequence, node A^T can be removed. The resulting structure is shown on the right of figure E2-30.

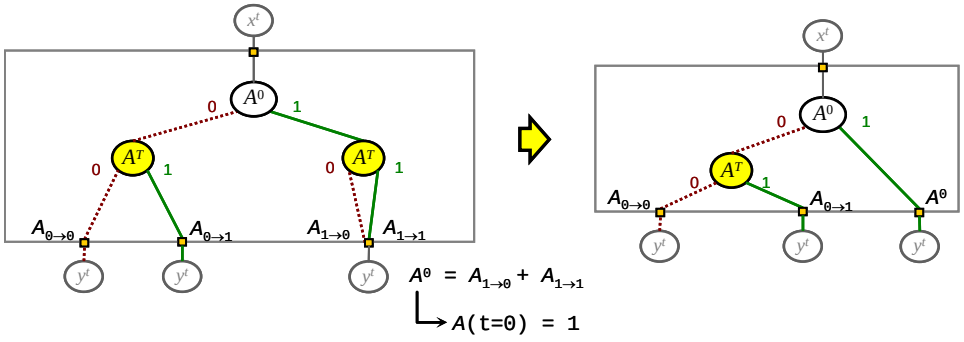


Figure E2-30: Example of a TFBDD reduction

As a result of the reductions, new paths result from the traversing of the new structures. The meaning of the new paths can be deduced analyzing the reduction procedure. For example, in figure E2-30, paths $A_{1\to0}$ and $A_{1\to1}$ end in the same node. The union of these two events is[65]:

$$\{[(A^0=1) \wedge (A^T=0)\,] \vee [(A^0=1) \wedge (A^T=1)]\} =$$
$$= \{[(A^0=1) \wedge [(A^T=0) \vee (A^T=1)]\} = \{A^0=1\}$$

[65] Remember that:
$A^0 \equiv A(t=0)$; $A^T \equiv A(t=T)$;

The formula can be written in a clearer way[66]:

$$A^0 \cdot \overline{A^T} + A^0 \cdot A^T = A^0 \cdot (\overline{A^T} + A^T) = A^0$$

Which is the meaning of traversing rightmost path of figure E2-30.

Figure E2-31 shows some other examples of TFBDD simplified structures.

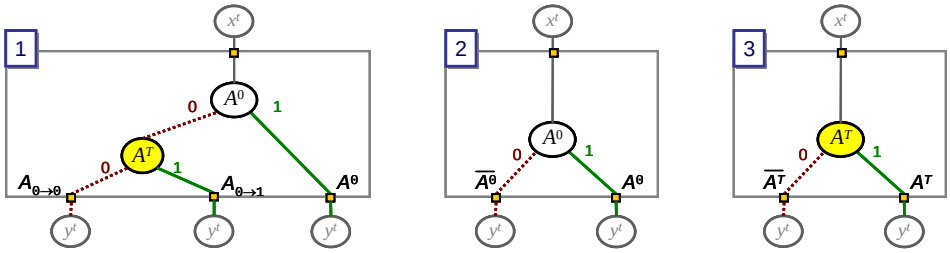


Figure E2-31: Some TFBDD simplified structures for signal A

The meaning of all possible paths that can be traversed in a signal's TFBDD structures has been represented in figure E2-32. For example, structure 1 of figure E2-31 has the first, second and fifth paths of paths listed in figure E2-32 (counted from left to right). Structure 3 of figure E2-31 has the two rightmost paths of figure E2-32.

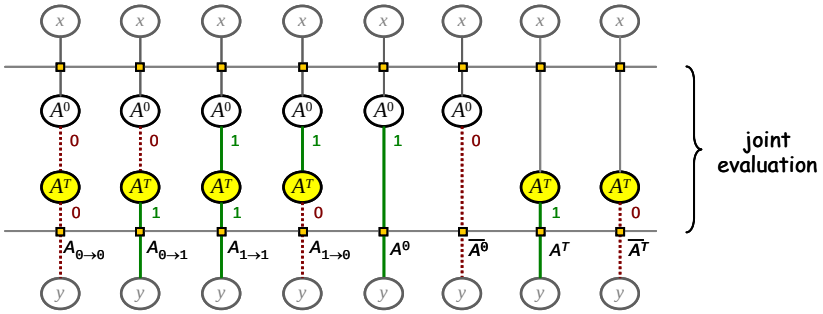


Figure E2-32: All possible paths that can be traversed in the TFBDD structures of a signal

After the TFBDD structure and interpretation has been exposed, the explanation about obtaining the activity function can be presented. Next subsection deal with this explanation and also describes how TFBDD are created.

E2.5.1.2 Obtaining the activity function from a TFBDD

The activity function has to be obtained bearing in mind that, for each path, the nodes of the same signal have to be jointly evaluated. Since input signals have been considered to be independent and the reconvergence analyses has been carried out, the signals in a TFBDD can be considered mutually independent. Therefore, the probability expression can be split.

An example may help to understand it. Figure E2-33 shows how to obtain the TFBDD of a multiplexer (figure E1-7 showed another similar example). The application of an XOR to the probability function at two consecutive times results in the TFBDD (the rightmost BDD of the figure).

[66] Remember that talking about events:

$A^0 \equiv \{A(t{=}0) = 1\}$; $A^T \equiv \{A(t{=}T) = 1\}$; $\overline{A^T} \equiv \{A(t{=}T) = 0\}$

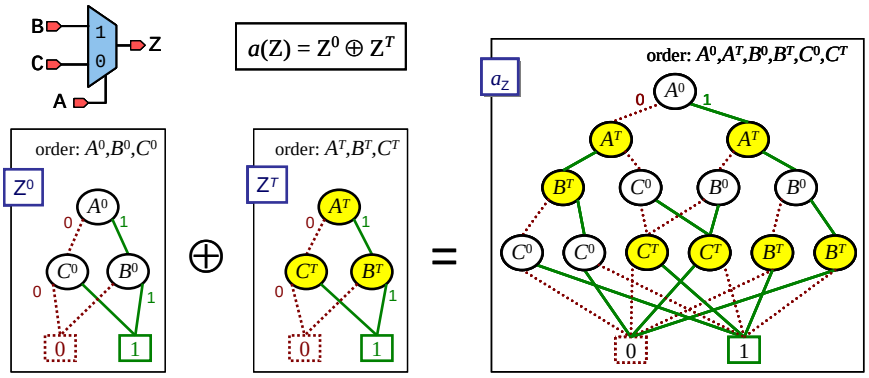


Figure E2-33: Obtaining the TFBDD of a multiplexer

Note that the TFBDD maintains the original BDD order but keeping together the variables of the same signal at different times. As a result, if the original ordering was {*A*, *B*, *C*} the new TFBDD ordering is {A^0, A^T, B^0, B^T, C^0, C^T}.

The steps to create the TFBDD are:

- For each BDD variable, a new variable of the same signal at time *T* is created
- Setting of the new variable order: The order among variables of different signals remains but, for variables of the same signal, the variable at time *T* follows the variable at time 0.
- Creation of a copy of the probability BDD but containing the variables related to time *T* (the original BDD is considered to be at time 0).
- Application of the XOR between the probability BDD at time 0 and the probability BDD at time *T*

Once the TFBDD is build, the activity function can be obtained traversing all the paths ending in the terminal node 1. Figure E2-34 shows all these paths for the TFBDD of figure E2-33. The meaning of each path has been written at its bottom. Note that in those paths where two nodes (at times 0 and *T*) of the same signal appear, these nodes are evaluated jointly.

Due to BDD properties, BDD paths are mutually disjoint. Therefore, the probability of each path can be calculated separately. Besides, because every BDD signal is assumed to be independent, probabilities can be separated inside each path. As a result, the activity function can be obtained as it is shown at the bottom of figure E2-34.

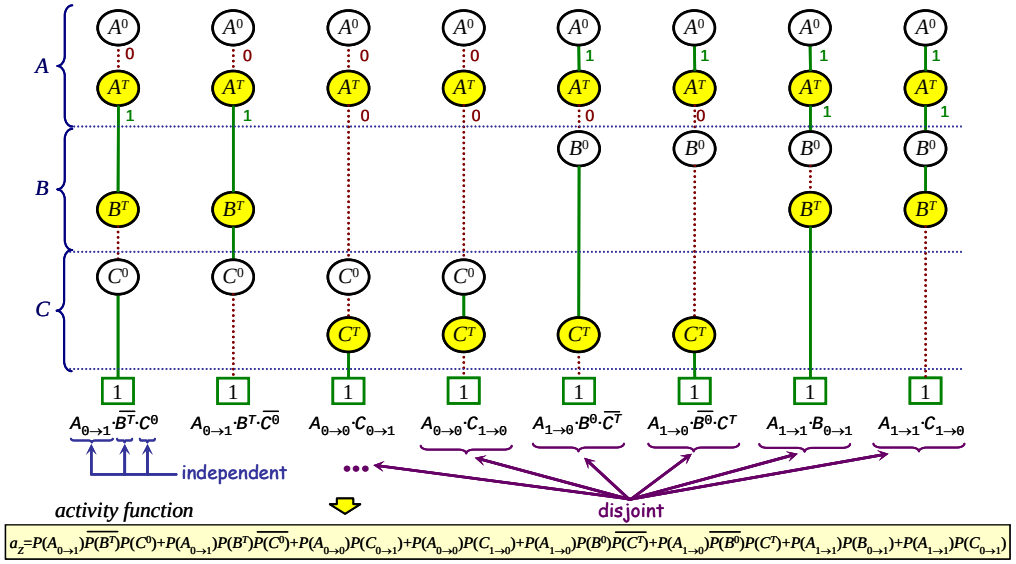


Figure E2-34: Traversing the paths ending in terminal node 1 to obtain the activity function (see the original TFBDD in figure E2-33

Nevertheless, it is not necessary to obtain the activity function to calculate the activity values. There are faster methods to calculate the activity values from the TFBDD. However, it has been explained in order to help understanding TFBDDs and to compare with the activity BDDs proposed in this Thesis, which will be expounded in the next section.

E2.5.2 Proposed activity BDD

With the purpose of reducing the activity BDD size, this Thesis proposes a new approach of representing them, whereby significant BDD size reductions are achieved.

Instead of using a timed based representation (x^0 and x^T), this Thesis proposes a representation based firstly on activity and inactivity (a_x and $\overline{a}_x$); and then on transition probability: $P(x_{0\to0})$, $P(x_{1\to1})$, $P(x_{0\to1})$, $P(x_{1\to0})$. This kind of representation has been called **activity BDD**, or in its short form ***a*BDD**.

In the next subsection the *a*BDD structure is described (§E2.5.2.1), which is an original proposal of this Thesis. Then, in subsection E2.5.2.2 the proposed *a*BDD structure is compared with the TFBDD structure. Following, a BDD activity operator will be defined. This activity operator definition is an original contribution of this Thesis and it is useful to create *a*BDDs from probability BDDs. Although several BDD operators exist, it is important to define an efficient operator since *a*BDDs are a new structure. Next, subsection E2.5.2.5 explains how to obtain the activity function from *a*BDDs.

Out of this section, sections E2.5.3 and E2.5.4 compare the resulting activity BDDs from different logic functions and number of inputs.

E2.5.2.1 *a*BDD structure

The general *a*BDD structure for any signal *A* is shown in figure E2-35. Instead of creating a new variable at time *T* (i.e. A^T), a new activity variable (a_A) is created[67]. In *a*BDDs, the variable order of different signals is maintained, but the new activity variables are set immediately before the variables of their same signal.

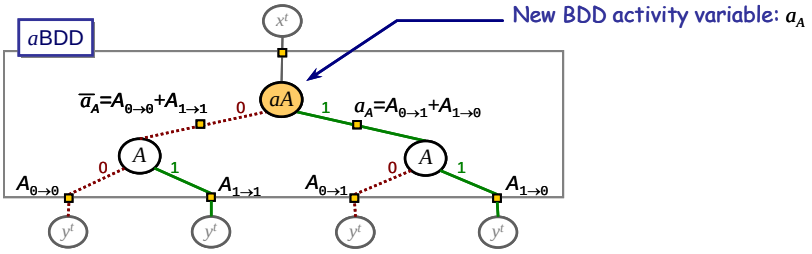


Figure E2-35: aBDD general structure for signal A

From figure E2-35 it can be seen that, the edge zero of the activity node represents the signal inactivity: $\overline{a}_A$. The other edge (edge 1) represents the signal activity: a_A. As previously seen with the equation of signal inactivity (equation E1.4):

$$\overline{a}_A = P(A_{0\to0}) + P(A_{1\to1})$$

And the equation of signal activity is (equation E1.3):

$$a_A = P(A_{0\to1}) + P(A_{1\to0})$$

When a signal node (*A*) follows its activity node (a_A), the edges of the signal node separate the above formulas in their terms. For example, if the edge zero of the activity node is taken, it represents the signal inactivity: $\overline{a}_A = P(A_{0\to0}) + P(A_{1\to1})$. Then, from the signal node *A*, if the edge 0 is taken, it means that the signal has no switching activity and it remains at zero. Therefore,

[67] In the figures, activity variable names are written without subindex (*aA*) in order to facilitate readability

this path represents the transition: $A_{0\to0}$. If, instead of taking this last edge, edge one were taken, it would also mean that signal *A* does not have activity, but the signal remains at one: $A_{1\to1}$.

On the contrary, if the edge one of the activity node is taken, it represents the signal activity $a_A = P(A_{0\to1}) + P(A_{1\to0})$. Then, taking the edge zero of the signal node it represents the transition $A_{0\to1}$. And the other edge represents the transition $A_{1\to0}$.

Similar to TFBDDs, *a*BDD structures can also be simplified. Examples of these simplified structures are shown in figure E2-36.

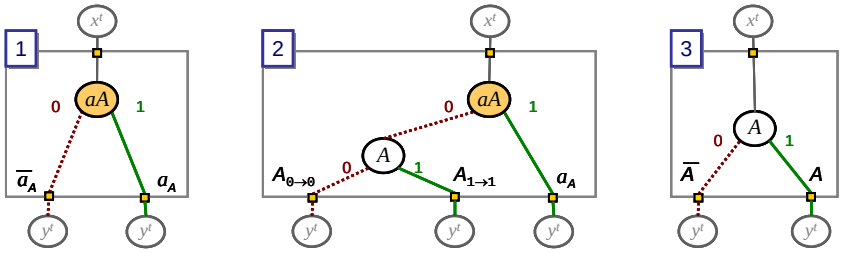


Figure E2-36: Some aBDD simplified structures for signal A

All the possible paths that can be found in *a*BDD structures of signal *A* are represented in figure E2-37. The meaning of each path has been written at their bottom. For example, the first structure of figure E2-36 contains the fifth and sixth paths of figure E2-37. The second structure of figure E2-36 contains the first, second and fifth paths of figure E2-37. And the third structure of figure E2-36 has the two rightmost paths of figure E2-37.

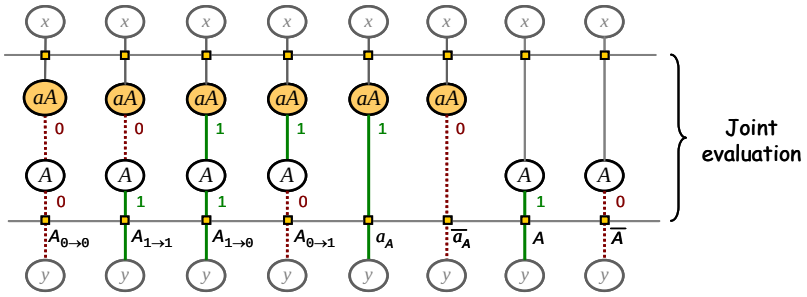


Figure E2-37: All possible paths that can be traversed in the aBDD structures of signal A

Equation E1.2 implies that transition probabilities from zero to one are equal to transition probabilities from one to zero: $P(A_{0\to1}) = P(A_{1\to0}) = \frac{1}{2}a_A$. As a consequence, more compact *a*BDD representations can be attained. Instead of taking the general structure of figure E2-35, which differentiates between $A_{0\to1}$ and $A_{1\to0}$, the new representation considers both edges as $\frac{1}{2}a_A$. Thus, the general structure looks like the one in figure E2-38. In this structure, the edges hanging from the signal node that comes from the edge one of the activity node are indistinct, since they both represent $\frac{1}{2}a_A$.

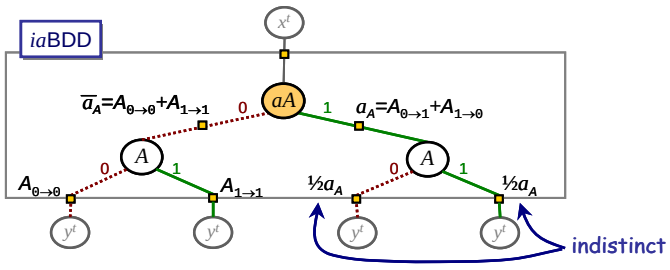


Figure E2-38: Indistinct aBDD general structure for signal A

This representation will be named *indistinct aBDD* (i*a*BDD), in contrast to the previous representation, which will be called *distinct aBDD* (d*a*BDD). Hereafter, whenever it is not specified whether an *a*BDD is distinct or indistinct, the *a*BDD is assumed to be indistinct (i*a*BDD).

Maintaining the conditions described in section E2.2, the usage of i*a*BDD instead of d*a*BDD does not produce different activity values because each signal's joint evaluation value is the same for both representations. Figure E2-39 shows the interpretation of all the possible paths of an indistinct *a*BDD. Note that all the paths have the same interpretation except for those two paths pointed with an arrow. These two paths are not distinct, but their numeric value is the same as to their equivalent d*a*BDD (compare with figure E2-37).

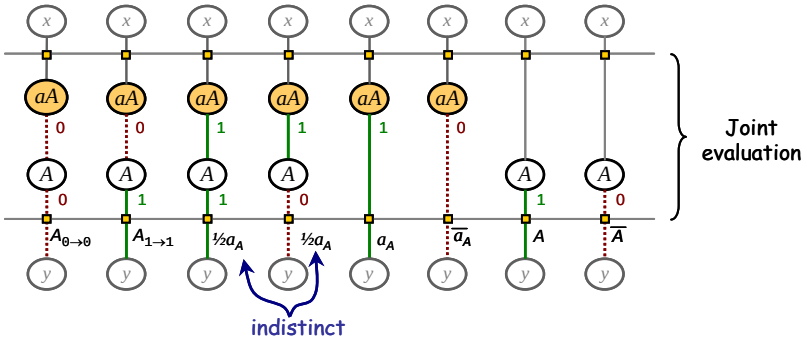


Figure E2-39: All possible paths that can be traversed in the indistinct aBDD structures of signal A

The usage of indistinct paths may considerably reduce the total number of BDD nodes due to their higher degree of freedom.

The usage of d*a*BDD may be interesting when is necessary to conserve the information about the kind of transition: 0→1 or 1→0. Even though the probability of these transitions is the same, they may be useful when input spatial correlations are considered.

Both indistinct and distinct *a*BDDs are original proposals of this Thesis. Nevertheless, in this Thesis, unless otherwise stated, only indistinct *a*BDDs will be used. The annex AII.4 analyses the differences between these *a*BDD (but it has not been translated to English)

E2.5.2.2 Comparison between *a*BDD and TFBDD structures

Due to the structural differences between *a*BDDs and TFBDDs, *a*BDDs achieve a considerable node reduction compared to TFBDDs.

There is no difference in the number of nodes between the general structures of *a*BDDs and TFBDDs (figures E2-28 and E2-35). However, the *a*BDD representation of many simplified structures is more advantageous than the TFBDD representation.

Figure E2-40 shows a very common situation of activity BDDs. In the TFBDD (on the left of figure E2-40), both paths related to transition 0→1 and transition 1→0 finish at the same node. The union of both transitions represents the switching activity, which precisely is the meaning of the edge one of the *a*BDD activity node. As a consequence, one node out of three is saved using *a*BDDs.

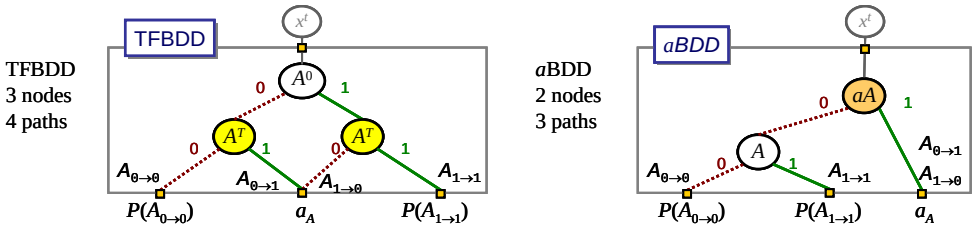


Figure E2-40: Comparison between TFBDD and aBDD structures when transitions 0→1 and 1→0 lead to the same node

Another even more advantageous situation happens when transitions 0→0 and 1→1 end in the same node, but different to the node in which transitions 0→1 and 1→0 finish. Transitions 0→0 and 1→1 represent signal inactivity, which is directly obtained by the edge zero of the activity

node. Figure E2-41 shows the difference between the TFBDD and the *a*BDD structures. In this case, two nodes out of three have been saved and two paths out of four.

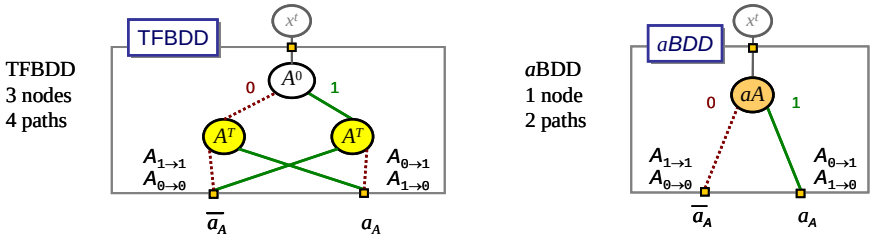


Figure E2-41: Comparison between TFBDD and aBDD structures when transitions can be reduced to activity and inactivity

To end, figure E2-42 shows how signal probability is represented in TFBDDs and *a*BDDs. If TFBDDs are used, two representations are possible: one for the probability at time 0 and another for the probability at time *T*. However, these probabilities are equal and they can be substituted by the signal probability without specifying time. That is exactly what *a*BDDs do. Hence, a node can be saved if both structures are used in TFBDDs.

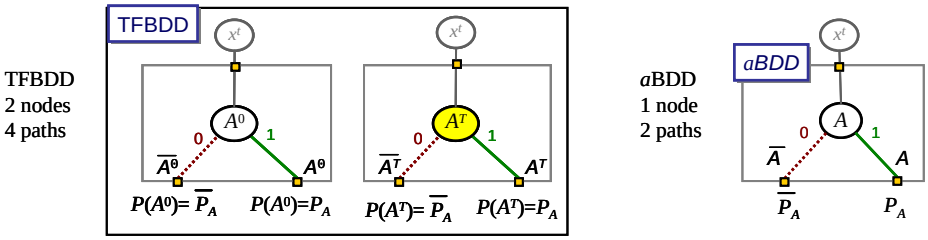


Figure E2-42: Comparison between the TFBDDs and aBDDs used to represent signal probability

There are more situations in which reductions are achieved; however, they will not be illustrated since they are variations of the already described situations.

It is important to remark that all the *a*BDD structures have a number of nodes equal or less than their corresponding TFBDD structures. Therefore, there will never be an *a*BDD larger than the TFBDD. In the practical examples, an important reduction has always been achieved (see sections E2.5.3, E2.5.4 and E3.4.1).

E2.5.2.3 Transforming TFBDDs into aBDDs

TFBDDs can be converted into *a*BDDs through a set elementary transformations based on the comparisons explained in previous section. Nevertheless, the inverse transformation, from *a*BDD to TFBDD, is not unique due to the *a*BDD simplifications that provoke an information loss. Under the conditions that have been stated in this Thesis (§E2.2), this information loss does not have any effect on the computation accuracy.

For example, the simplification carried out on figure E2-42 implies an irreversible information loss such that, once the *a*BDD is obtained, it is not possible to know from which TFBBD structure it came.

The transformation from TFBDDs to *a*BDDs will be explained through an example. The original TFBDD will be an OR gate TFBDD, which is the one showed in figure E1-7.

The original OR gate TFBDD is on the left of figure E2-43. An arrow points to a temporal node B^0. Neither on the top nor at the bottom of this node there is another node of the same signal *B*. Therefore, it forms itself a TFBDD basic structure like the one on the left of figure E2-42. The basic structure has been inscribed in a square in the middle TFBDD. In consequence, in the same way as in figure E2-42, it can be transformed into a timeless node, such as the TFBDD on the right of the figure.

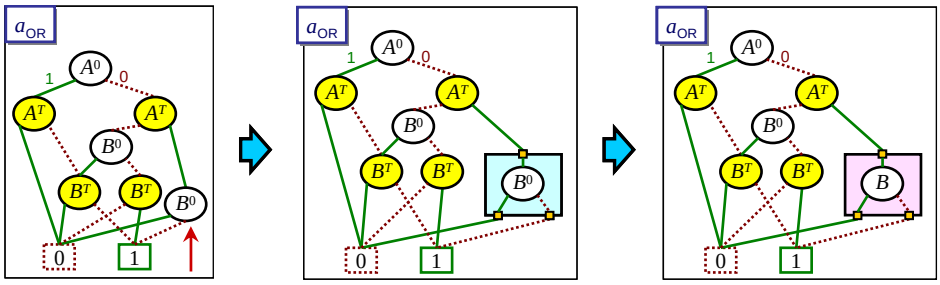


Figure E2-43: Step 1, transforming a temporal node into a timeless node

Starting from the TFBDD of the right of figure E2-43, the next step is to duplicate the node B^T that is being pointed in the left BDD of figure E2-44. Because this node can be accessed from two edges, this step is carried out in order to independently analyze the paths leading to the node. The result of this step is shown in the right BDD of figure E2-44, where the two new nodes have been inserted in a rounded rectangle.

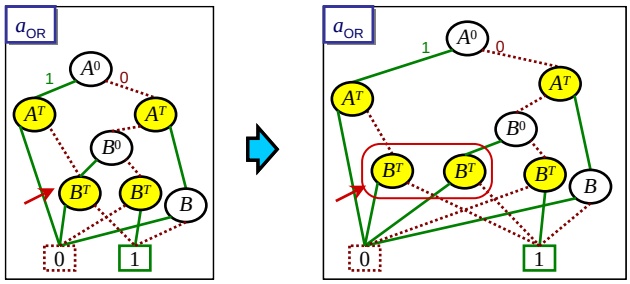


Figure E2-44: Step 2, splitting node B^T in two nodes

One of the resulting nodes of the previous step can be transformed in a timeless node. The node has been pointed in the leftmost BDD of figure E2-45. It is the same transformation that has been carried out in figure E2-42. The result of this transformation is shown in the rightmost BDD of figure E2-45

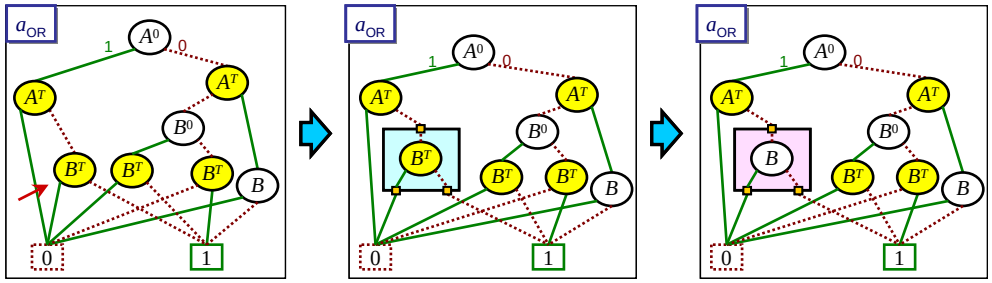


Figure E2-45: Step 3, transforming a temporal node into a timeless node

The resulting node of the previous transformation and the resulting node of figure E2-43 are equivalent[68]. Hence, these nodes can be jointed in one node. These two nodes have been pointed in the left BDD of figure E2-46. The right BDD of figure E2-46 shows the result of this node reduction.

[68] Two nodes are equivalent if they not only are nodes of the same variable, but also, the entire structures from those nodes to the terminal nodes are equal.

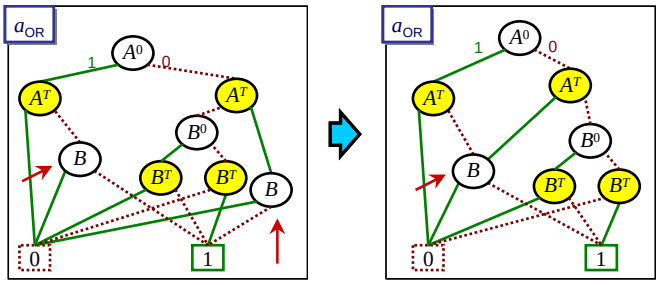


Figure E2-46: Step 4, reduction of equivalent nodes

A set of three nodes have been marked on the left BDD of figure E2-47. As it can be observed from the middle BDD of figure E2-47, these three nodes forms the same TFBDD structure as the one of figure E2-41. Therefore, the structure can be substituted for an activity node. The resulting BDD is the rightmost BDD of figure E2-47.

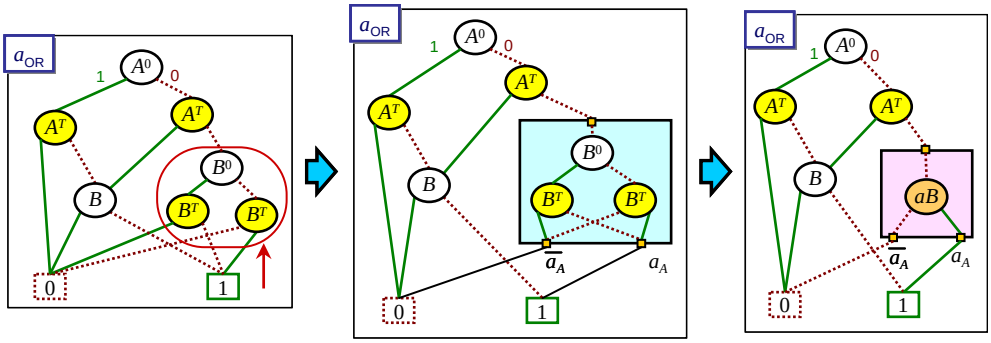


Figure E2-47: Step 5, TFBDD structure substitution for an activity node

The last transformation corresponds to nodes of signal *A*. These nodes have been marked on the leftmost BDD of figure E2-48. In the middle BDD, it can be observed that the TFBDD structure of the three nodes corresponds to the structure shown in figure E2-40. Consequently, the TFBDD structure has been substituted by its equivalent *a*BDD structure. The rightmost BDD is the resulting *a*BDD.

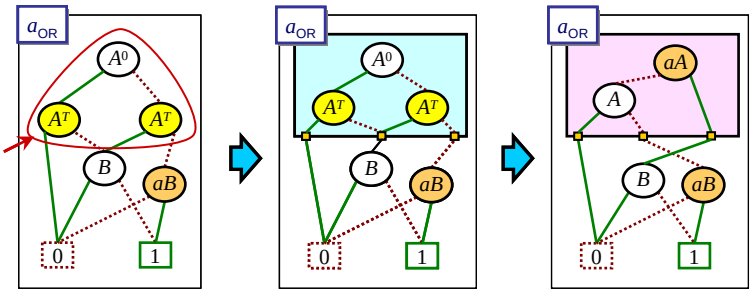


Figure E2-48: Step 6, TFBDD structure substitution

As a summary of the transformations and in order to compare both activity BDDs, figure E2-49 shows both representations. As it can be seen, the *a*BDD is simpler, having 4 nodes against the 7 nodes that the TFBDD has. There is also an important reduction of the number of paths. The TFBDD has 9 paths and the *a*BDD has 5 paths. The figure also compares which paths end in terminal node one and in terminal node 0. In the TFBDD, four paths end in terminal node one, whilst in the *a*BDD only two paths end in terminal node one.

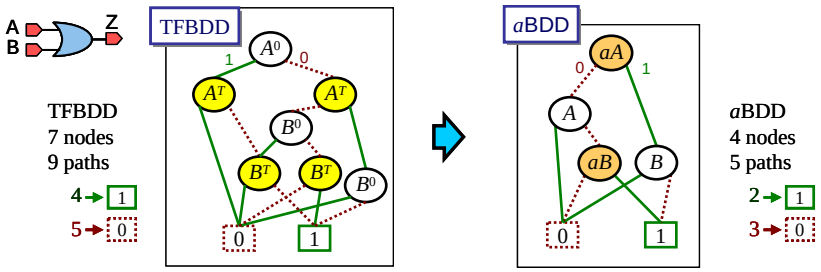


Figure E2-49: Comparison between the TFBDD and the aBDD of an OR gate

The transformation from TFBDD into *a*BDD has been implemented as a function in the library package BuDDy [64] and also it has been integrated in Ardid [132]. Consequently, both representations of a circuit activity BDDs can be automatically compared.

E2.5.2.4 Definition of an activity operator

In the previous section, the transformation from TFBDDs into *a*BDDs was explained. However, it is not very useful to propose an optimized BDD representation (*a*BDD) if the previous representation (TFBDD) is needed to obtain the proposed one. Therefore, an activity operator has been defined in this Thesis to create *a*BDDs directly from the probability BDDs, without using TFBDDs.

There are operators defined for BDDs, for example: AND, OR, composition,... Since a new BDD representation has been defined, an operator to build it has to be defined. Once the operator is defined, it can be implemented as a function in a library package for manipulating BDDs.

The first step to create the *a*BDD is the variable ordering definition. The *a*BDD order is the same as the probability BDD order, but the signal activity variables are just before their corresponding signal variables. That is, if the BDD variable order is {*A*, *B*, *C*} the *a*BDD variable order is $\{a_A, A, a_B, B, a_C, C\}$.

Only the original probability BDD is needed to create the *a*BDD. It is important to remark that the activity operator is defined for just one operand but, because it is a recursive function, when traversing the BDD, in the subsequent calls to the function it may have different operands. Therefore, it is no sense to apply the activity operator to different operands, but it makes sense to call the function with different arguments when traversing the BDD.

Consequently, the activity operator is defined for two operands, which are probability BDDs: $\mathcal{M}$ and $\mathcal{N}$ ([69]). The result of the operator is an activity BDD: $a\mathcal{Q}$ ([70]):

$$a[\mathcal{M},\mathcal{N}] \rightarrow a\mathcal{Q}$$

Therefore, the domain of the function is formed by all the probability BDDs and the image (or range) of the function are the *a*BDDs.

When the function only has one operand ($\mathcal{M}$), the operation is carried out over the same operand:

$$a[\mathcal{M}] = a[\mathcal{M},\mathcal{M}] \qquad \text{(E2.7)}$$

Since it is not easy to use formulas to represent BDDs, in order to facilitate the comprehension, figures representing the operations will also be used. Equation E2.7 will be represented as figure E2-50 shows. The operands are between brackets and the operands are probability BDDs.

[69] Probability BDDs will be written in this kind of cursive font: $\mathcal{M}$

[70] Activity BDDs (*a*BDD) will be written in the same kind of cursive font, but they will be preceded by letter "*a*". For example: $a\mathcal{Q}$

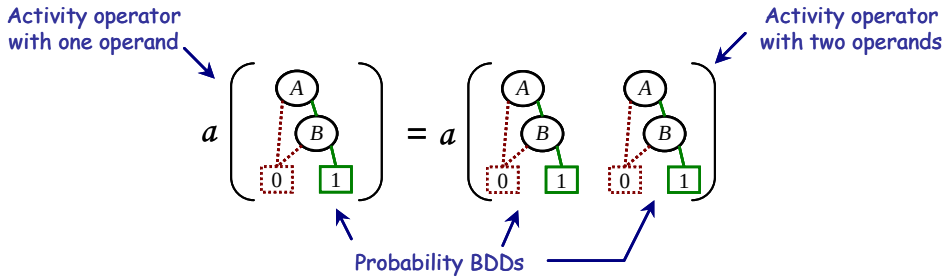


Figure E2-50: Activity operator equivalence for one operand

The activity operator is commutative:

$$a[\mathcal{M},\mathcal{N}] = a[\mathcal{M},\mathcal{N}]$$

It is no sense to analyze the associative and distributive properties of the activity operator because the domain of the function (probability BDDs) is different than the image (*a*BDD). Therefore, the result of the operation cannot be operated again.

The activity operator applied to a constant BDD (*0* or *1*) is the *a*BDD zero:

$$a[0] = a[1] = 0 \tag{E2.8}$$

Then, also:

$$a[0,0] = a[1,1] = 0 \tag{E2.9}$$

Equations E2.8 and E2.9 are represented in figure E2-51:

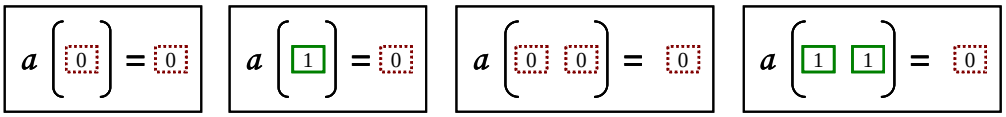


Figure E2-51: Activity operator applied to constant and equal BDDs

The activity equation applied to constant but different BDDs equals the *a*BDD one:

$$a[0,1] = a[1,0] = 1 \tag{E2.10}$$

Equation E2.10 can be represented as shown in figure E2-52:

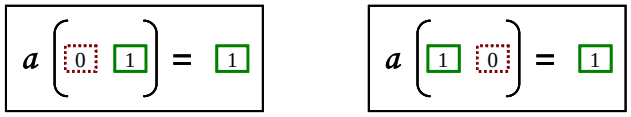


Figure E2-52: Activity function applied to constant but different BDDs

BDD one and BDD zero belongs to both the domain (probability BDD) and the image (*a*BDD).

The activity operator applied to a simple[71] node results in the activity node of the signal of the original node.

$$a[\mathcal{A}] = a\mathcal{A} \tag{E2.11}$$

Being $\mathcal{A}$ a simple probability BDD of signal *A*, and $a\mathcal{A}$ the *a*BDD of signal *A*. To clarify it, figure E2-53 shows the representation of equation E2.11.

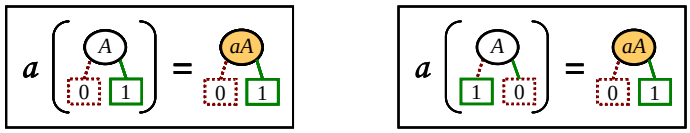


Figure E2-53: Activity operator applied to simple probability BDDs

[71] A simple node is a node whose edges end in terminal nodes but different nodes. That is, one edge leads to terminal node 1 and the other leads to terminal node 0

The activity operator applied to simple nodes of variable *A*, but being one one negated, results in the inactivity of *A*.

$$a[\mathcal{A}, \overline{\mathcal{A}}] = \overline{a\mathcal{A}} \tag{E2.12}$$

Where $\overline{\mathcal{A}}$ is a simple probability BDD that is the negated form of $\mathcal{A}$. That is, it represents $\overline{P}_A$. While $\overline{a\mathcal{A}}$ is the *a*BDD that represents the inactivity of signal *A*. Equation E2.12 has been represented in figure E2-54:

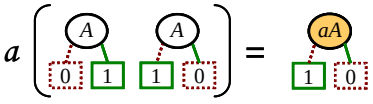


Figure E2-54: Activity function applied to simple probability BDDs of the same signal but one of them negated

The activity function applied to a BDD and the terminal node zero results in the same BDD:

$$a[\mathcal{M}, 0] = \mathcal{M} \tag{E2.13}$$

The activity function applied to a BDD and the terminal node one results in the negated BDD:

$$a[\mathcal{M}, 1] = \overline{\mathcal{M}} \tag{E2.14}$$

These two properties embrace equations E2.9 and E2.10; as a consequence, terminal node zero can be considered as the neutral element of the activity operator. Equations E2.13 and E2.14 have been represented in figure E2-55

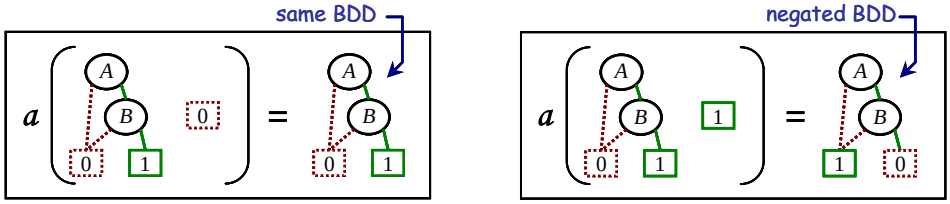


Figure E2-55: Activity function applied to a BDD and the BDD zero and one

The activity function applied to BDDs with **different variables in their root nodes** is executed in a recurrent way through the child nodes. Let $\mathcal{M}$ and $\mathcal{N}$ be the operands, and assuming that the variable of the root node of $\mathcal{M}$ precedes the variable of the root node of $\mathcal{N}$ in the BDD ordering. Then:

- The *a*BDD root node is created with the same variable as the root node of $\mathcal{M}$ (the variable that goes first in the BDD order)
- Attached to edge zero of the *a*BDD root node is the result of the activity function applied to $\mathcal{N}$ and the cofactor of $\mathcal{M}$ in respect to the negated variable of the root node
- Attached to edge one of the *a*BDD root node is the result of the activity function applied to $\mathcal{N}$ and the cofactor of $\mathcal{M}$ in respect to the variable of the root node

What can be formulated in the following equation:

$$a[\mathcal{M}, \mathcal{N}] = \text{ITE}\left\{\text{root}(\mathcal{M}) \, , \, a[\mathcal{M}_{\text{root}(\mathcal{M})}, \mathcal{N}] \, , \, a[\mathcal{M}_{\overline{\text{root}(\mathcal{M})}}, \mathcal{N}]\right\} \tag{E2.15}$$

Where ITE(*A*, $\mathcal{R}$, $\mathcal{S}$) is the function *if-then-else* for BDDs. The first argument *A* of the function ITE is a BDD variable, while the second and third arguments ($\mathcal{R}$ and $\mathcal{S}$) are BDDs. Function ITE builds a BDD creating the root node containing the variable of the first argument (*A*). Then, the second argument ($\mathcal{R}$) is attached to the edge one of the root node. And the third argument ($\mathcal{S}$) is attached to the edge zero of the root node.

Going back to equation E2.15, function root($\mathcal{M}$) obtains the variable of the root node of $\mathcal{M}$. Then, $\mathcal{M}_{\text{root}(\mathcal{M})}$ is the cofactor of $\mathcal{M}$ in respect to the variable of its root node. And $\mathcal{M}_{\overline{\text{root}(\mathcal{M})}}$ is the cofactor of $\mathcal{M}$ in respect to the negated variable of its root node.

Equation E2.15 may be hard to understand; hence, in order to facilitate its comprehension, figure E2-56 shows its graphical representation. In the figure, the activity function is applied to two BDDs ($\mathcal{M}$ and $\mathcal{N}$) that have root nodes of different variable. The variable of the root node of $\mathcal{M}$ is *A*, which precedes *B*, the variable of the root node of $\mathcal{N}$. Therefore, variable *A* is taken as the first argument of function ITE. In both edges, the activity function is applied again. One of the arguments is $\mathcal{N}$, and the other argument is the cofactor of $\mathcal{M}$ in respect to the *A*, negated or not, depending if it is the edge zero or one.

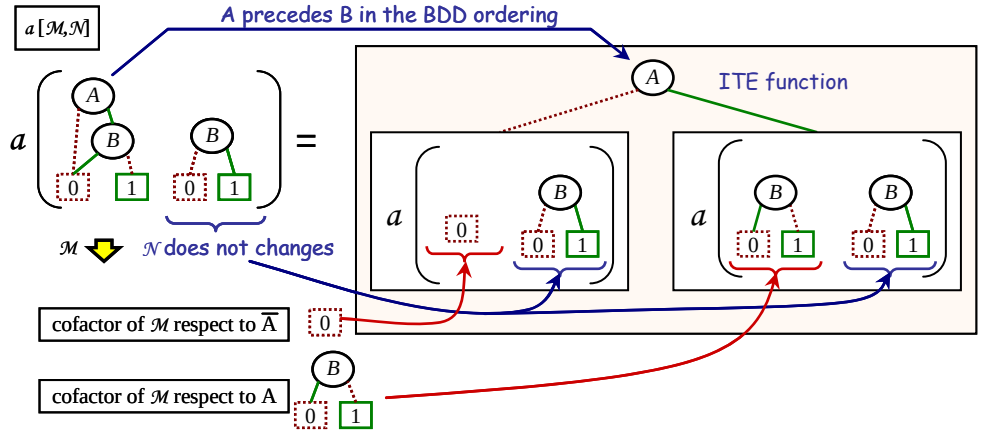


Figure E2-56: Activity function applied to BDDs with root nodes of different variable

The next step can be calculated from what has been already explained about the activity operator. From the BDD of figure E2-56, the activity equation is applied to both edges. The BDD of edge zero is obtained from equation E2.13 (figure E2-55). The BDD of edge one is obtained from equation E2.12 (figure E2-54). This step has been sketched in figure E2-57, where resulting *a*BDD is shown on the right of the figure.

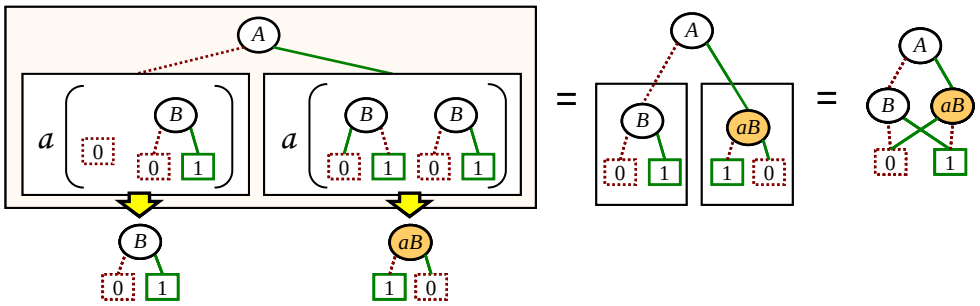


Figure E2-57: Continuation of the aBDD construction of figure E2-56

Last, the activity function applied to BDDs with **the same variable in their root nodes** will be explained. First, the general *a*BDD structure is created for the signal of the root node (figure E2-38). The application of the activity function to some combinations of the cofactors of the operands will be attached to the four branches of the general structures.

It will be explained with the example of figure E2-58. The operand of the left is $\mathcal{M}$ and the operand of the right is $\mathcal{N}$. The four cofactors ($\mathcal{M}_A$, $\mathcal{M}_{\bar{A}}$, $\mathcal{N}_A$, $\mathcal{N}_{\bar{A}}$) in respect to the variable of the root node (*A*) have been drawn in the figure.

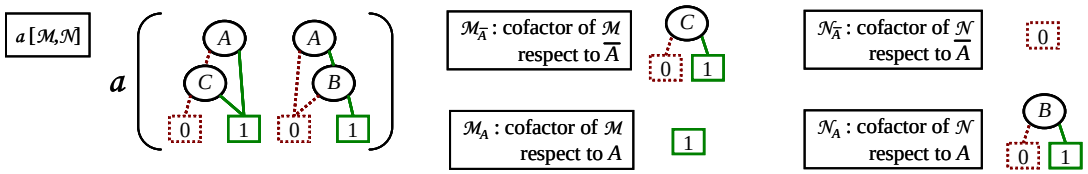


Figure E2-58: Cofactors of the operands in respect to the variable of the root node

The application of the activity function to the four cofactor combinations has been drawn in figure E2-59. In the figure, the four combinations have been tagged. Tag 0→0 corresponds to

$a[\mathcal{M}_{\overline{A}}, \mathcal{N}_{\overline{A}}]$, that is, the resulting BDD from the activity function applied to the cofactors in respect to the negated variable of the root node ($\overline{A}$). Tag 1→1 corresponds to $a[\mathcal{M}_A, \mathcal{N}_A]$, that is the resulting BDD from the activity function applied to the cofactors in respect to the variable of the root node (A). Tag 1→0 corresponds to $a[\mathcal{M}_A, \mathcal{N}_{\overline{A}}]$ and the tag 0→1 corresponds to $a[\mathcal{M}_{\overline{A}}, \mathcal{N}_A]$

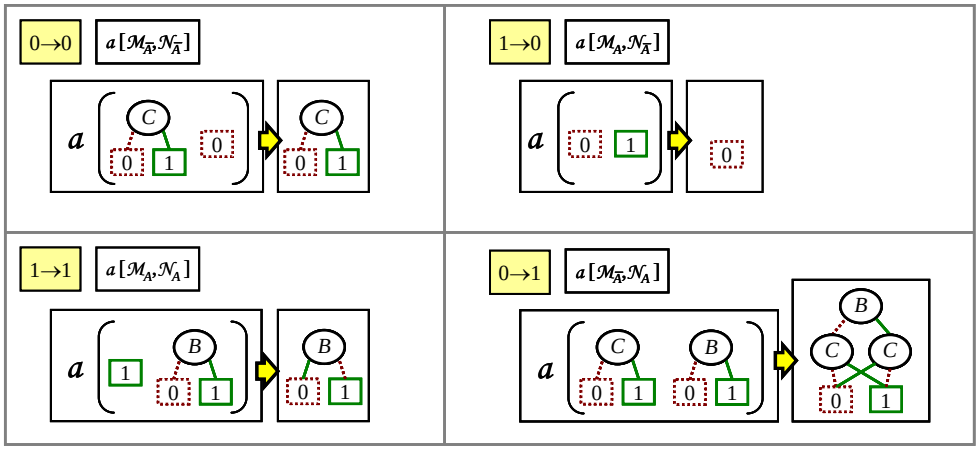


Figure E2-59: Results of the activity function applied to the cofactors

These four BDDs will be attached to the four edges of the general *a*BDD structure created for variable *A*. The BDDs will be attached to the edges according to their tags. The BDD of tag 0→0 will be attached to the edge 0→0 and the BDD of tag 1→1 will be attached to the edge 1→1.

It is usual for the resulting BDDs of tags 0→1 and 1→0 to be equivalent. But it does not always happen, as in the example of figure E2-59. When they are different, there are two ways to arrange them. If an indistinct *a*BDD (§E2.5.2.1) is being created, the smaller[72] BDD has to be attached to the edge zero of the activity edges. This could form smaller *a*BDDs (see anex AII.4, in Spanish). In this example, the BDD of tag 1→0 would be attached to the edge zero. Figure E2-60 shows this process: The BDD of tag 1→0 is smaller that the BDD of tag 0→1. Therefore BDD 1→0 is attached to the edge zero of the activity edges (those marked with $½a_A$). The resulting i*a*BDD has been drawn on the right of figure E2-59.

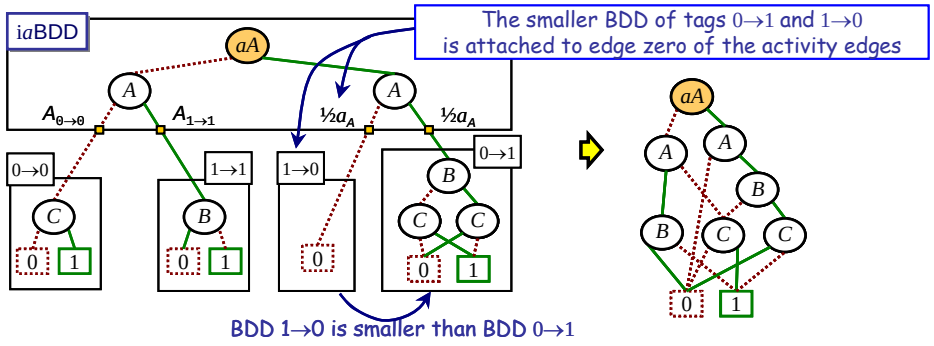


*Figure E2-60: Obtaining the ia*BDD of the example of figure E2-58

To build a distinct *a*BDD, the BDD 0→1 has always to be attached to edge zero. Figure E2-61 shows how the d*a*BDD would be built.

[72] In the BuDDy package, BDDs are defined by an unique value, which ususally is related to its size. In this case, small is refferred to this value, though it may larger in size.

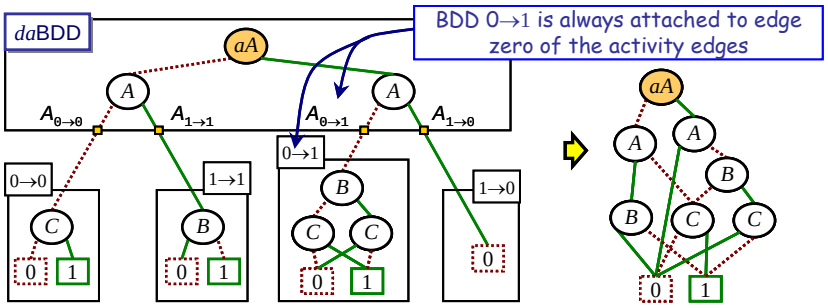


Figure E2-61: Obtaining the daBDD of the example of figure E2-58

Due to the d*a*BDD arrangement, the activity function is not commutative when building d*a*BDDs.

To conclude this section, it will be remarked the importance of the activity operator definition in order to create *a*BDDs independently from TFBDDs, that is, just using probability BDDs. All the definitions and properties permit the univocal creation of BDDs with the defined characteristics in section E2.5.2.1. It has also favored the process automation and the integration in a BDD manipulation computer program.

E2.5.2.5 Getting the activity equation from *a*BDDs

Obtaining the activity equation is not a necessary step to the activity calculations. There are more efficient methods to calculate activity from BDDs (see annex AI.2 in Spanish). However, explaining the process may be useful in order to understand BDDs and to compare *a*BDDs with TFBDDs.

The obtaining of the activity equation from an *a*BDD follows a similar process to the obtaining of the probability equation from a probability BDD (§2.2.2, figure 2-9 in Spanish), or the activity equation from a TFBDD (§E2.5.1.2, figure E2-34).

To obtain the activity equation from an *a*BDD, all the paths finishing in terminal node one have to be traversed. Figure E2-62 shows the example of an OR gate *a*BDD.

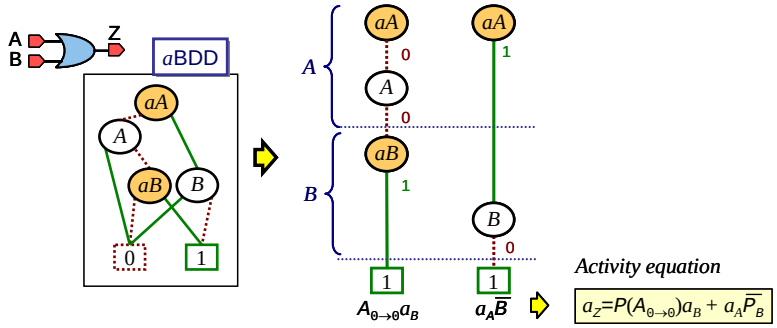


Figure E2-62: Obtaining the activity equation from an OR gate aBDD

Therefore, the obtained activity equation is:

$$a_{OR} = a_A \cdot \overline{P_B} + P(A_{0\to0}) \cdot a_B \qquad \text{(E2.16)}$$

The OR gate activity equation obtained from a TFBDD was explained in figure E1-7:

$$a_{OR} = P(A_{1\to0})\,\overline{P_B^T} + P(A_{0\to1})\,\overline{P_B^0} + P(A_{0\to0})\,P(B_{0\to1}) + P(A_{0\to0})\,P(B_{1\to0}) \qquad \text{(E2.17)}$$

Equations E2.16 and E2.17 are equivalent. Considering equation E1.1 (stationary process):

$$\overline{P_B^T} = \overline{P_B^0}$$

Then, equation E2.17 can be converted into:

$$a_{OR} = [P(A_{1\to0}) + P(A_{0\to1})]\,\overline{P_B} + P(A_{0\to0})\,P(B_{0\to1}) + P(A_{0\to0})\,P(B_{1\to0})$$

Grouping the last two terms and knowing that $a_x=P(x_{1\to0})+P(x_{0\to1})$, (equation E1.3), the equation becomes identical to the one obtained using the *a*BDD (equation E2.16):

$$a_{OR} = a_A \cdot \overline{P}_B + P(A_{0\to0}) \cdot a_B$$

In consequence, the *a*BDD representation does not imply any accuracy loss. Instead, it benefits from the stationary assumption and from the transition probability equality: $P(x_{1\to0})+P(x_{0\to1})$. Besides, because the representation is based on activity and inactivity, further node reductions are obtained.

E2.5.3 Comparison between TFBDDs and *a*BDD of various logic functions

This section compares the *a*BDD and TFBDD sizes of various logic gates and functions.

Figure E2-49 showed the difference between the TFBDD and the *a*BDD of an OR gate. Since the activity of a signal is equal to the activity of the same signal being negated, the result is identical for a NOR gate.

In the next subsection, the differences between TFBDDs and *a*BDDs will be analyzed for gates AND, NAND, XOR, XNOR and for multiplexers.

E2.5.3.1 AND & NAND gates

The activity BDD of gates AND & NAND are similar to the OR gate. They have the same node arrangement. The only change is related to the edges. Therefore, as it can be seen in figure E2-63, the OR gate TFBDD has the same number of nodes as the AND gate TFBDD. Likewise, both *a*BDDs have the same number of nodes. Thus, a 43% in node reduction and a 44% in path reduction is achieved.

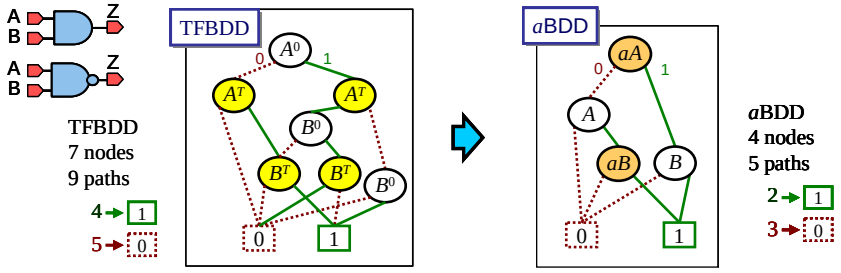


Figure E2-63: Comparison between the TFBDD and aBDD of AND& NAND gates

E2.5.3.2 XOR & XNOR gates

As it can be observed from figure E2-64, compact representations are achieved using *a*BDDs for XOR and XNOR gates. The number of nodes has been reduced by 57% using *a*BDDs instead of TFBDDs. The number of paths has been reduced by a quarter.

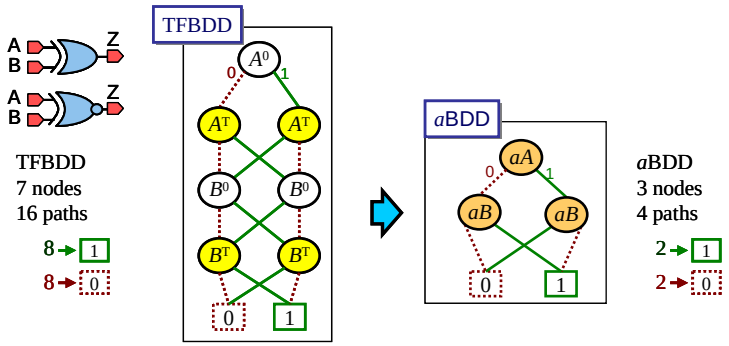


Figure E2-64: Comparison between the TFBDD and aBDD of XOR & XNOR gates

E2.5.3.3 Multiplexer

Important node reductions are also attained in multiplexers by the use of *a*BDDs. Figure E2-65 shows both representations. The number of nodes has been reduced by 46% using *a*BDDs instead of TFBDDs. The number of paths has been reduced by a half.

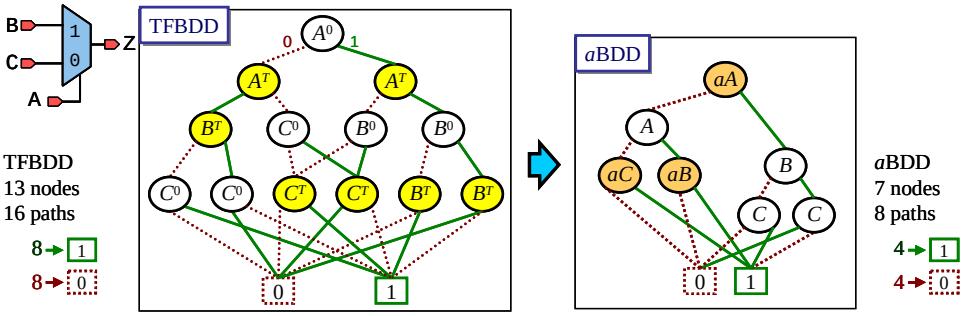


Figure E2-65: Comparison between the TFBDD and aBDD of a multiplexer

E2.5.4 BDD sizes concerning the number of inputs

This section analyzes the increment in the number of nodes and paths of the probability and activity BDDs considering the number of inputs of the logic functions. The analysis has been made for AND gates, XOR gates, and for multiplexers. Remember that the gates NAND, OR, NOR have similar results to the AND gate and the XNOR gate has similar results to the XOR gate.

E2.5.4.1 AND gate

Figure E2-66 shows the BDDs of a 3-input AND gate. The probability BDD has 3 nodes and 4 paths. The TFBDD has 11 nodes and 16 paths. And the *a*BDD has 7 nodes and 9 paths. Therefore, the number of nodes has been reduced by 36% using *a*BDDs instead of TFBDDs. And the number of paths has been reduced by 44%.

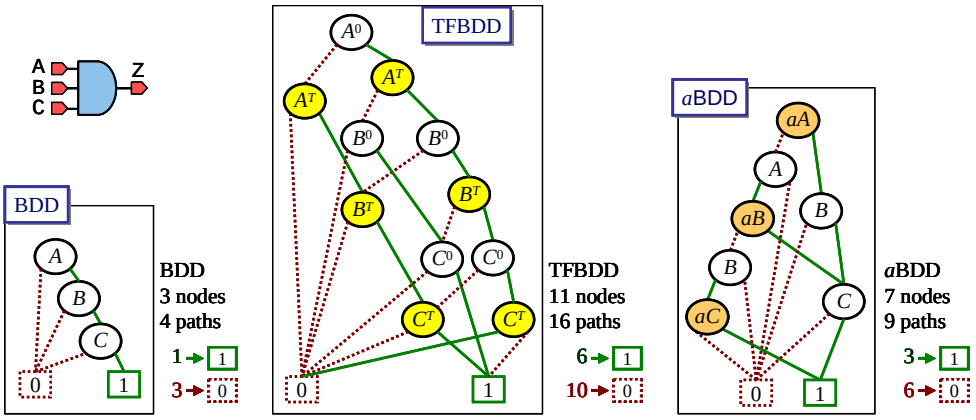


Figure E2-66: BDD, TFBDD and aBDD of a 3-input AND gate

Figure E2-67 shows the effect of adding another input to the AND gate. Now, the number of nodes of the TFBDD is 15, while the number of nodes of the *a*BDD is 10. Then, the number of nodes has been reduced by 33% and the number of paths also by 44%.

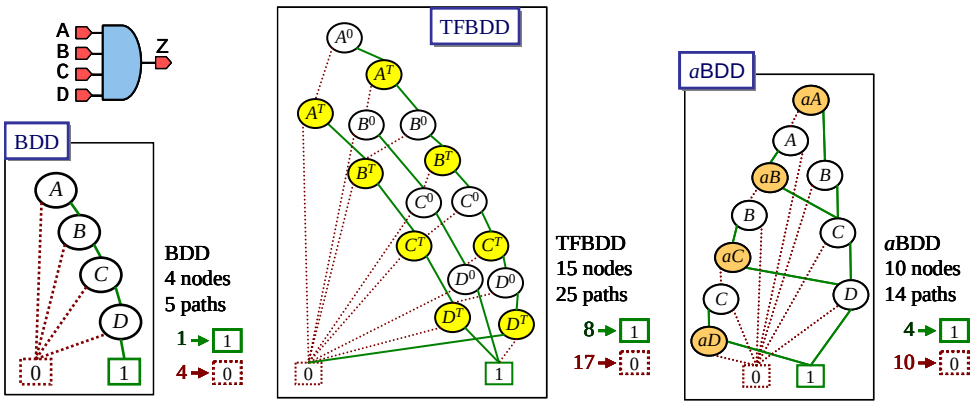


Figure E2-67: BDD, TFBDD and aBDD of a 4-input AND gate

From the previous results, the data of larger AND gates can be inferred. Figure E2-68 shows the BDDs of a 5-input AND gate.

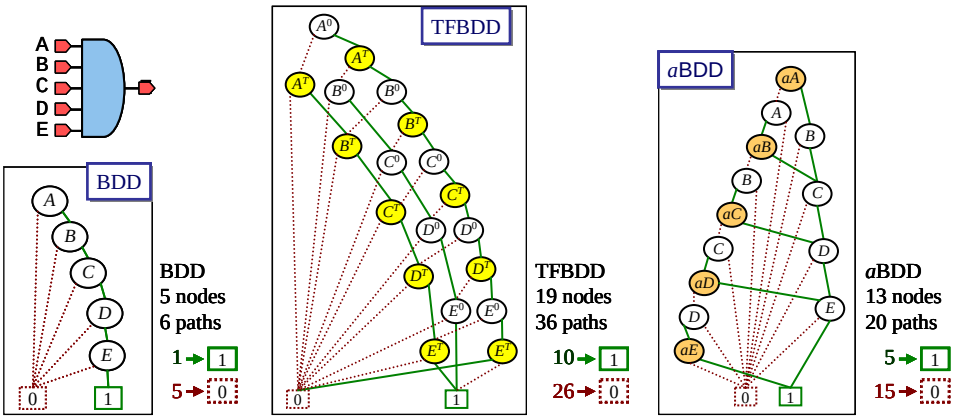


Figure E2-68: BDD, TFBDD and aBDD of a 5-input AND gate

The BDD node increase with the number of inputs of the AND gate has been presented in table E2-1. For the three kinds of BDDs, the number of nodes follows an arithmetic progression, whose n^{th}-terms has been included at the bottom of the table. When the number of inputs tends to infinite, the usage of *a*BDDs reduces he number of nodes by 25%.

Number of inputs of the AND gate	Number of nodes			node reduction TFBDD→*a*BDD
	BDD	TFBDD	*a*BDD	
2	2	7	4	43%
3	3	11	7	36%
4	4	15	10	33%
5	5	19	13	32%
6	6	23	16	30%
n	n	4n-1	3n-2	$25\%|_{n\to\infty}$

Table E2-1: Number of nodes of the BDDs of an AND gate depending on the number of inputs

Table E2-2 shows the data related to the number of paths. The table includes the numbers of paths going to terminal node one (*paths to 1*), the number of paths going to terminal node zero (*paths to 0*) and the total number of paths. That is, including both paths ending in terminal nodes one and zero (*total paths*).

Number of inputs of the AND gate	paths to 1			path to 0			total paths			total reduction TFBDD→*a*BDD
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	
2	1	4	2	2	5	3	3	9	5	44%
3	1	6	3	3	10	6	4	16	9	44%
4	1	8	4	4	17	10	5	25	14	44%
5	1	10	5	5	26	15	6	36	20	44%
6	1	12	6	6	37	21	7	49	27	45%
n	1	2n	n	n	n^2+1	$½(n^2+n)$	n+1	n^2+2n+1	$½(n^2+3n)$	$50\%|_{n\to\infty}$

Table E2-2: Number paths of the BDDs of an AND gate depending on the number of inputs

E2.5.4.2 XOR gate

This subsection analyzes the effect of adding inputs to an XOR gate. An XOR gate with multiple inputs is equivalent to split the XOR operation in 2-inputs XOR, as figure E2-69 shows.

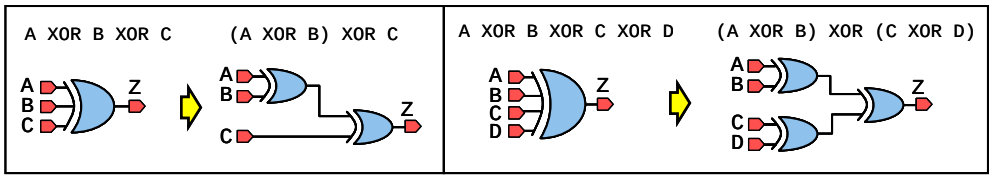


Figure E2-69: 3-input and 4-input XOR equivalence

Figure E2-70 shows the BDDs of a 3-input XOR gate. The probability BDD has 5 nodes and 8 paths. The TFBDD has 11 nodes and 64 paths. And the *a*BDD has 5 nodes and 8 paths. Therefore, the TFBDD has more than double the nodes the *a*BDD has, and four times the number of paths.

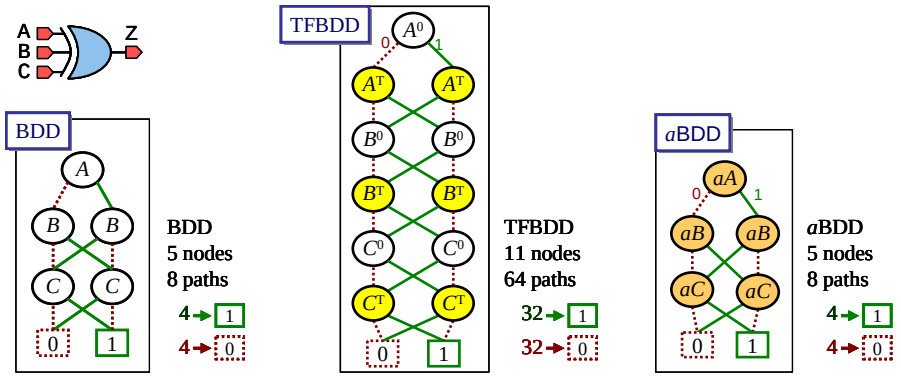


Figure E2-70: BDD, TFBDD and aBDD of a 3-input XOR gate

Figure E2-71 shows the BDDs of a 4-input XOR gate. The BDD and the *a*BDD have the same number of nodes and paths (7 nodes and 16 paths). In contrast, the TFBDD has more than twice the *a*BDD nodes and the square of the *a*BDD paths.

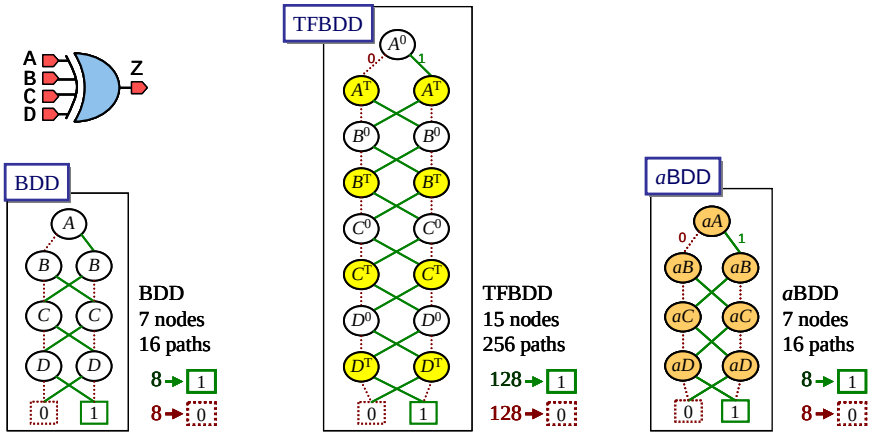


Figure E2-71: BDD, TFBDD and aBDD of a 3-input XOR gate

Table E2-3 shows the XOR BDD size related to its number of inputs. BDDs and *a*BDDs follow an arithmetic progression with a common difference of 2, whereas the TFBDD follow an arithmetic progression with a common difference of 4.

Number of inputs of the XOR gate	Number of nodes			node reduction TFBDD→*a*BDD
	BDD	TFBDD	*a*BDD	
2	3	7	3	57%
3	5	11	5	54%
4	7	15	7	53%
5	9	19	9	53%
6	11	23	11	52%
n	2n-1	4n-1	2n-1	$50\%\vert_{n\to\infty}$

Table E2-3: Number of nodes of the BDDs of an XOR gate depending on the number of inputs

Table E2-4 shows the data related to the number of paths.

Number of inputs of the XOR gate	paths to 1			path to 0			total paths			total reduction
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	TFBDD→*a*BDD
2	2	8	2	2	8	2	4	16	4	75%
3	4	32	4	4	32	4	8	64	8	87%
4	8	128	8	8	128	8	16	256	16	94%
5	16	512	16	16	512	16	32	1024	32	97%
6	32	2048	32	32	2048	32	64	4096	64	98%
n	2^{n-1}	2^{2n-1}	2^{n-1}	2^{n-1}	2^{2n-1}	2^{n-1}	2^n	2^{2n}	2^n	$100\%\|_{n\to\infty}$

Table E2-4: Number paths of the BDDs of an XOR gate depending on the number of inputs

E2.5.4.3 Multiplexer

In this subsection the BDD size of multiplexer is analyzed. In order to perform an independent analysis from the partition method, the analyzed multiplexers are not partitioned.

Distinct *a*BDD (d*a*BDD) and indistinct *a*BDD (i*a*BDD) are analyzed because there are differences related to their size that did not exist in the previous analysis. As it was explained in section E2.5.2 (and in the annex AII.4, in Spanish), the indistinct *a*BDD is the one that is being used in this Thesis by default.

The probability BDD of a 2-alternative multiplexer was shown in figure E2-33. The TFBDD and the *a*BDD were compared in section E2.5.3.3. There are no size differences between the d*a*BDD and the i*a*BDD for this multiplexer.

If a third alternative is added to this multiplexer, a new selection signal (bit) has to be added. The resulting multiplexer is not complete, in the sense that there is a condition that does not assign any alternative, or one alternative is assigned to two different conditions.

Figure E2-72 shows the BDDs of a 3-alternative multiplexer. Alternative *D* is assigned when selection signal *A*1 is one, independently from the other selection signal value (*A*0). Using *a*BDDs the number of nodes has been reduced by 40% and the number of paths by the half. The d*a*BDD and the i*a*BDD have the same number of nodes and paths; thus, they have not been distinguished in the figure.

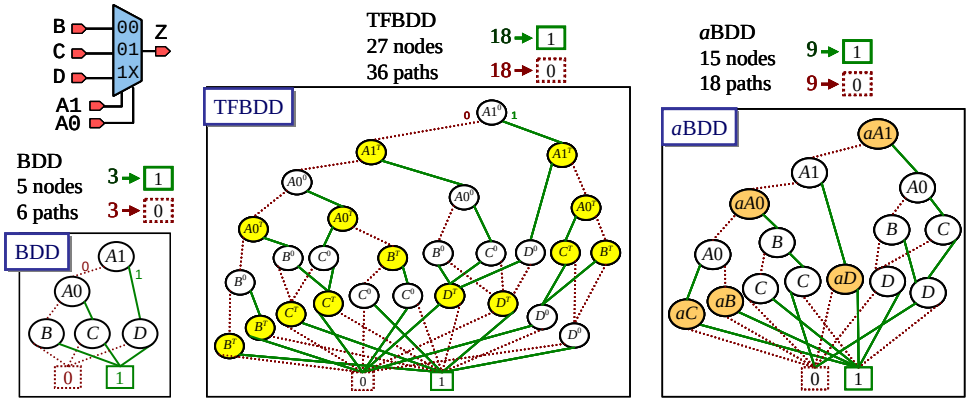


Figure E2-72: BDD, TFBDD and aBDD of a 3-alternative multiplexer

A 4-alternative multiplexer is complete, that is, there are no alternatives that are selected by more than one condition. Figure E2-73 shows the probability BDD and the TFBDD of the multiplexer. Figure E2-74 shows both the i*a*BDD and the d*a*BDD. From the BDDs of the figures, the significant growth of the activity BDDs in respect to the probability BDD can be appreciated. Comparing activity BDDs, note the important node reduction achieved by the i*a*BDD: a reduction by over 44%.

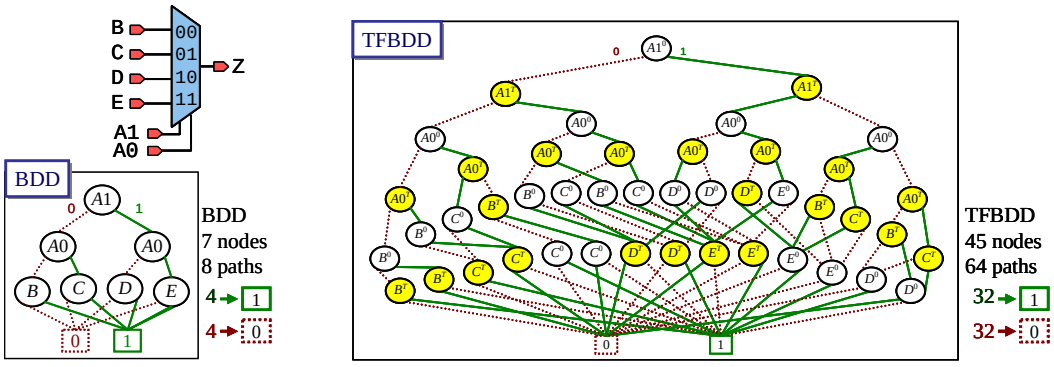


Figure E2-73: BDD and TFBDD of a 4-alternative multiplexer

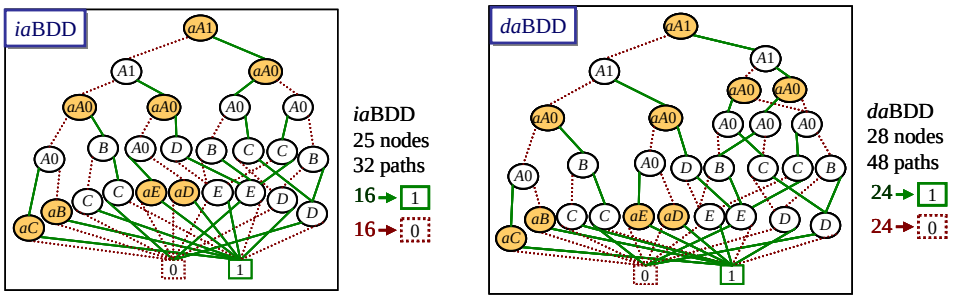


Figure E2-74: iaBDD and daBDD of a 4-alternative multiplexer

The size of incomplete[73] multiplexers depends on how the alternatives are distributed; therefore, there is not a unique BDD for a specific number of alternatives. As an example, figure E2-75 shows the BDD size of two different configurations of a 6-alternative multiplexer. The multiplexer on the left has smaller BDDs because of its more homogeneous alternative distribution.

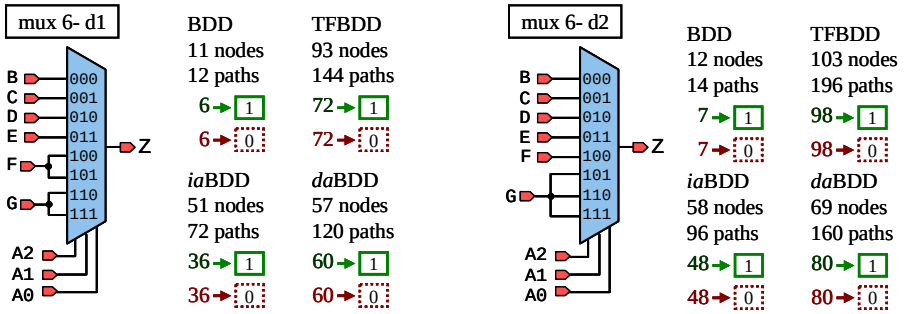


Figure E2-75: BDD sizes of two different configurations of a 6-alternative multiplexer

Due to these variations in BDD size within a specific number of alternatives, it is difficult to extract a BDD growth sequence. Therefore, when the multiplexer reaches 8 alternatives, only a number of alternatives that are to the power of two has been included in table E2-5.

The node reduction by using *ia*BDDs reaches 50% with a large number of alternatives.

[73] With a number of alternatives different to a power of two

Multiplexer alternatives	BDD	TFBDD	i*a*BDD	d*a*BDD	reduction TFBDD→i*a*BDD	reduction TFBDD→d*a*BDD
2	3	13	7	7	46%	46%
3	5	27	15	15	44%	44%
4	7	45	25	28	44%	38%
5	9	67	37	40	45%	40%
6 -d1	11	93	51	57	45%	39%
6 -d2	12	103	58	69	44%	33%
7	13	123	67	78	46%	37%
8	15	157	85	103	46%	34%
16	30	530	283	367	47%	31%
32	63	2173	1117	1477	49%	32%
64	127	8445	4285	5773	49%	32%
128	255	33277	16765	22813	50%	31%
256	511	132093	66301	90685	50%	31%
n	2n-1	~2(n+1)²	~(n+1)²	~(4/3)(n+1)²	$50\%\|_{n\to\infty}$	$31\%\|_{n\to\infty}$

Table E2-5: Number of nodes of the BDDs of a multiplexer depending on the number of alternatives

The variation in the number of paths is shown in table E2-6.

Multiplexer alternatives	Total number of paths				reduction TFBDD→i*a*BDD	reduction TFBDD→*da*BDD
	BDD	TFBDD	*ia*BDD	*da*BDD		
2	4	16	8	8	75%	75%
3	6	36	18	18	50%	50%
4	8	64	32	48	50%	25%
5	10	100	50	66	50%	34%
6 -d1	12	144	72	120	50%	17%
6 -d2	14	196	96	160	50%	18%
7	14	196	98	162	50%	17%
8	16	256	128	224	50%	12%
16	32	1024	512	956	50%	7%
32	64	4096	2048	3968	48%	3%
64	128	16384	8192	16128	49%	2%
128	256	65536	32768	65024	50%	1%
256	512	262144	131072	261129	50%	0%
n	2n	4n²	2n²	$4n^2\|_{n\to\infty}$	$50\%\|_{n\to\infty}$	$0\%\|_{n\to\infty}$

Table E2-6: Number of paths of the BDDs of a multiplexer depending on the number of alternatives

To conclude this section, it is important to remark the significant reduction achieved by the usage of *a*BDDs instead of TFBDDs. These reductions goes from 25% for AND & OR gates to 50% for multiplexers and XOR gates. The number of paths is reduced even more.

E2.6 RTL ordering of BDDs

The variable order of BDDs has a critical influence on their size [13], [52], [78], [112]. For example, depending on the variable order, the BDD size of an *n*-bit adder can vary from $O(n)$ to $O(2^{n/2})$ [78]. Consequently, it is important to arrange BDDs with an adequate ordering.

Nevertheless, getting an optimum order is an NP-Hard problem about which much research has been carried out [52].

For a circuit with *n* inputs, there are *n*! possible orderings. Therefore, for large circuits, not only the BDD size may be a problem, but also determining the appropriate ordering.

This Thesis proposes to make use of the RTL information to choose a favorable ordering. Usually, signals at RTL are arranged with a particular order. Thus, considering the *RTL natural order* would consist in taking the signals as they appear when traversing the EHM (§E2.3.2). Besides, at RTL there is information related to operators, operands and the signals grouped as vectors. This information allows to order the signals depending on their vector indexes and the operations the signals are in.

These ordering capabilities – those related to the signal arrangement and those related to the RTL information about operators and vectors – are no easy to obtain at gate level.

Following, the guidelines to obtain the *RTL natural order* will be given. These guidelines have been obtained in an empirical way. Thus, the guidelines will be validated with simple examples. The experimental results chapter will corroborate these results with larger circuits. The order related to the vector indexes and operands will be also demonstrated in a practical way in the experimental results chapter (§E3).

The proposed guidelines are:

1. Signals within conditions of higher hierarchy (outer conditional statements) precede signals of statements affected by those conditions (see example 1, figure E2-77)
2. Therefore, conditions of higher hierarchy are followed by the subsequent conditions until the assignments affected by the conditions.
3. Once a branch has been taken, all its child branches are taken until the sub-tree have been covered before any other branches and sub-trees are taken.
4. Therefore, a signal of an assignment statement may precede a signal of a condition if the condition does not affect that assignment statement
5. Reconvergent signals will be the last signals of their reconvergence region (see example 2, figure E2-79)
6. Generally, multiplexer alternatives being more complex, having greater depth or having a larger amount of dependences are traversed first.
7. The same happens to a logic gate: the input that is more complex, has larger depth or has a larger amount of dependences is traversed first (see example 5, figure E2-85)
8. When rules 6 or 7 have been taken, the reconvergent signals affecting both alternatives (or inputs) can be set at the end of the complex alternative (or input), or they can be set at the end of their whole reconvergence area (see examples 6 y 8)
9. The less complex alternative (or input) may also be taken, but if there is a great difference in respect to the number of final dependences between the alternatives (or inputs), the reconvergence signals are not set at the end of the reconvergence area. Instead, they are set when the appear (see example 5, figure E2-84)

These guidelines are open, that is, more than one ordering is possible. These rules are easy to follow if the circuit is analyzed with the EHM. On the contrary, they are hard to observe at gate level since it is difficult to know which signals are conditions. Thus, usually, at gate level an arbitrary order is taken, which may be later optimized.

These guidelines have been inferred without the development of a theory that lays its foundations. Nevertheless, the proposal seems logical because the rules fit in the BDD structure, as it can be seen in the next subsection. The following subsection shows examples of the application of the guidelines and the resulting BDDs are compared with all possible orderings.

E2.6.1 RTL ordering examples

In the following subsections examples of the RTL ordering will be shown. In these examples no disjoint partitions will be performed in order to compare the orderings related to the same circuit areas.

E2.6.1.1 Example 1

The first example is a simple conditional statement that is equivalent to a 2-alternative multiplexer (figure E2-76). If the signal within the condition is taken first, the BDD structure is a representation of a conditional statement. That is, when the order is *ABC*, root node *A* selects alternative *B* or *C* depending on its value (if *A*=1 alternative *B* is selected; whilst, if *A*=0 alternative *C* is selected). But, if any other signal is taken as the first, that signal is going to be evaluated before knowing if it has been selected, what usually provokes an increase of the nodes.

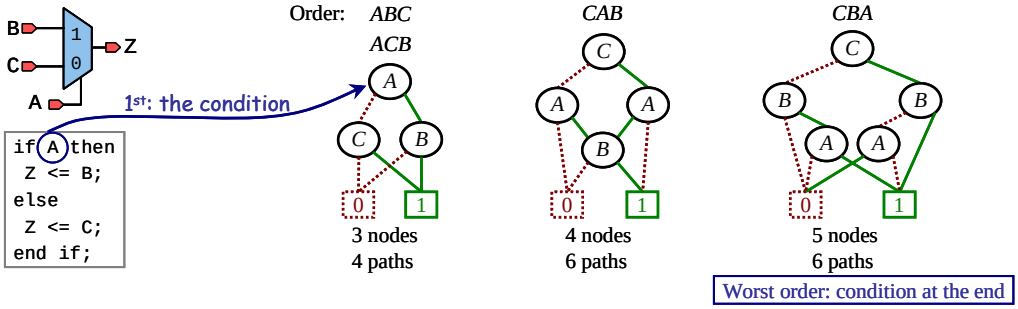


Figure E2-76: BDDs of a 2-input multiplexer depending on the variable order

In order to help the understanding how the ordering is chosen, figure E2-77 shows the EHM and the route taken to obtain the ordering. Starting at output signal *Z*, the conditional statement is reached (represented by a rhombus). In a conditional statement, the condition has to be taken prior to the alternatives (rule 1). Therefore, signal *A* goes first in the variable ordering. Afterwards, there are two possibilities: to take the path zero (doted line, on the left of the figure) or to take the path one (solid line, on the right of the figure). In this example, both routes lead to an optimum ordering: *ACB* and *ABC*.

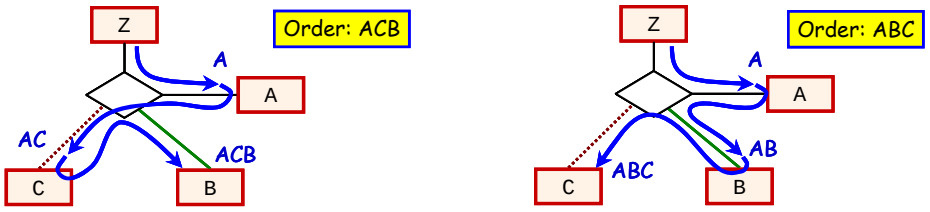


Figure E2-77: Routes through the EHM to obtain the RTL variable order of a multiplexer

The variable order also influences on the activity BDDs: a good probability BDD order results in a good activity BDD order. Table E2-7 shows the number of nodes and paths of BDDs, TFBDDs and *a*BDDs depending on the variable order.

Note how using *a*BDDs with the RTL ordering the number of nodes can be reduced from 21 for the worst case TFBDD to 7 nodes for the RTL ordered *a*BDD. That is, the nodes have been reduced to a third.

Order	Nodes			Paths		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
ABC - ACB	3	13	7	4	16	8
BAC - CAB	4	18	11	6	36	15
BCA - CBA	5	21	12	6	36	17

Table E2-7: BDD size of a multiplexer depending on the variable order

E2.6.1.2 Example 2

The circuit analyzed in this example is shown in figure E2-78, which is the same circuit of figure E2-18. Because the circuit has 4 inputs, there are 24 possible variable orderings. In order to give an idea about the size of the resulting BDDs, figure E2-78 shows four of the 24 orderings. The leftmost BDD is one of the BDDs with the lowest number of nodes (5 nodes). The first variable in this BDD is signal *A*, which is the signal of the condition. The last variable is the reconvergent signal. Therefore, it is one of the BDDs that would be built following the proposed RTL ordering guidelines.

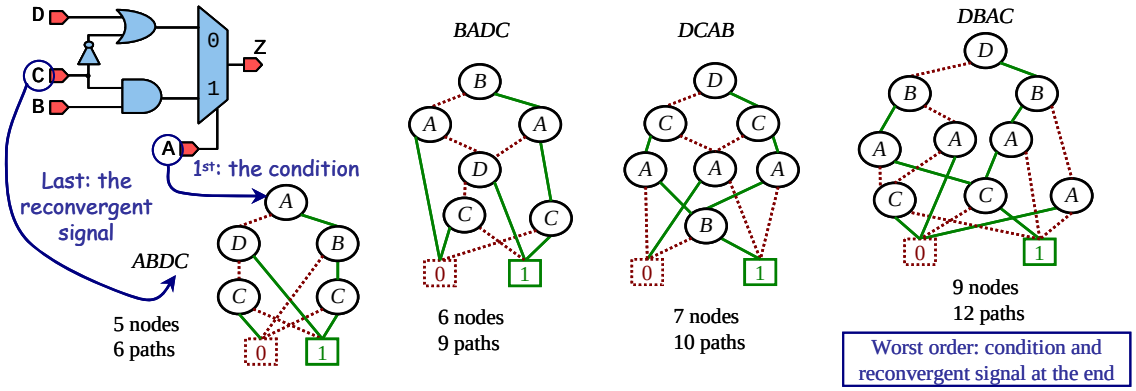


Figure E2-78: Some probability BDDs of example 2 for different variable orderings

Figure E2-79 shows the routes through the EHM to obtain the RTL order. From signal *Z*, the conditional statement is reached (rhombus). From the conditional statement, the condition always has to be taken first. Therefore, signal *A* would be the first variable in the RTL ordering. Then, since the alternative paths have similar complexity, any of them could be taken. Figure E2-79 shows both alternatives. Due to the reconvergence of signal *C*, in both cases it should be left in the last position.

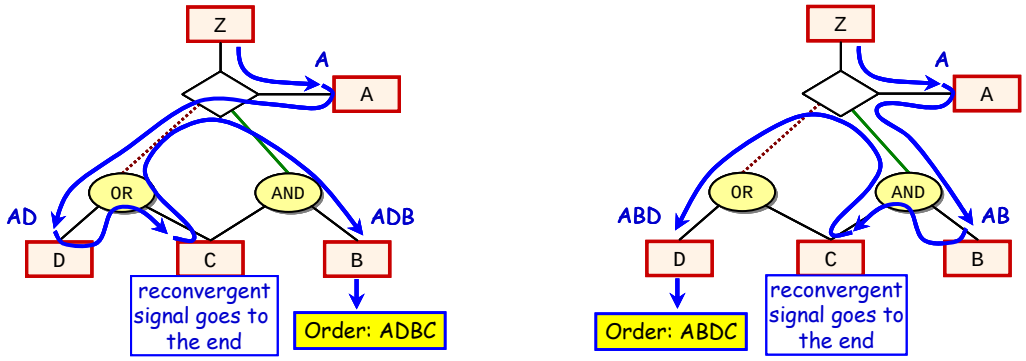


Figure E2-79: Routes through the EHM to obtain the RTL variable order of example 2

Table E2-8 shows the number of nodes and paths of BDDs, TFBDDs and *a*BDDs depending on their variable order. The RTL orderings have the minimum BDD size. They are on the top of the table, highlighted in boldface type: *ABDC* and *ADBC*.

Order	Nodes			Paths		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
ABDC - ADBC	5	21	13	6	36	17
CABD - CADB	5	27	15	6	36	18
ABCD - ADCB	5	25	16	6	36	19
ACBD - ACDB	5	27	16	6	36	19
BADC - DABC	6	26	17	9	81	34
CBDA - CDBA	6	28	17	8	64	25
BACD - DACB	6	30	20	9	81	38
CBAD - CDAB	6	31	21	8	64	31
BCDA - DCBA	7	33	21	10	100	37
BCAD - DCAB	7	36	25	10	100	47
BDCA - DBCA	8	45	27	12	144	51
BDAC - DBAC	9	53	32	12	144	53
Mean	6,3	31,8	20	8,5	76,8	32,4
Median	6	29	18,5	8,5	72,5	32,5

Table E2-8: BDD size of example 2 depending on the variable order

The mean and the median have been included at the bottom of the table. Supposing that the arbitrary gate level order results similar to the median order, the usage of the RTL order would reduce the number of nodes by almost 30% (from 18.5 to 13 comparing between *a*BDDs). Adding the reductions related to the RTL order and the usage of *a*BDDs instead of TFBDDs, the number of nodes goes from 29 (median order with TFBDD) to 13 (RTL order with *a*BDD). That is, a node reduction of more than the half.

E2.6.1.3 Example 3

This circuit has two conditional statements. The circuit corresponds to the reconvergence area of the circuit of figure E2-20. Figure E2-80 shows the circuit and some of the orderings.

The leftmost BDD is one of the BDD with the lowest number of nodes (6). The first variable in this BDD is signal *A*, which is the signal of the condition. The signal of the inner condition (*M*) has to be before its alternatives (*D* and *E*). Following these rules will lead to a minimum size BDD.

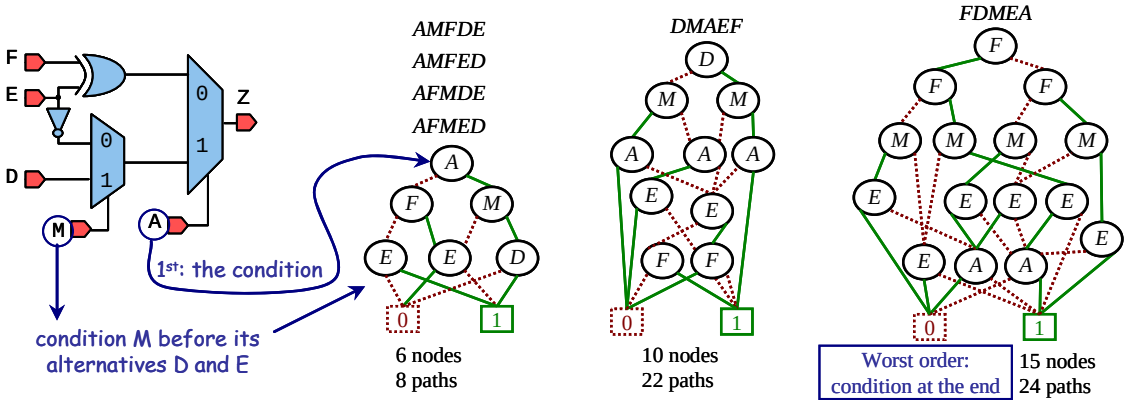


Figure E2-80: Some probability BDDs of example 3 for different variable orderings

Figure E2-81 shows the routes through the EHM to obtain the RTL order. From signal Z, the conditional statement is reached (rhombus). From the conditional statement, the condition always has to be taken first. Therefore, signal A would be the first variable in the RTL ordering. Then, any of the alternatives could be taken, but the reconvergence signal (*E*) should be left to

the end (rule 5). When the inner conditional statement is taken, the conditional also has to be taken first. Therefore, signal *M* precedes signals *D* and *E*.

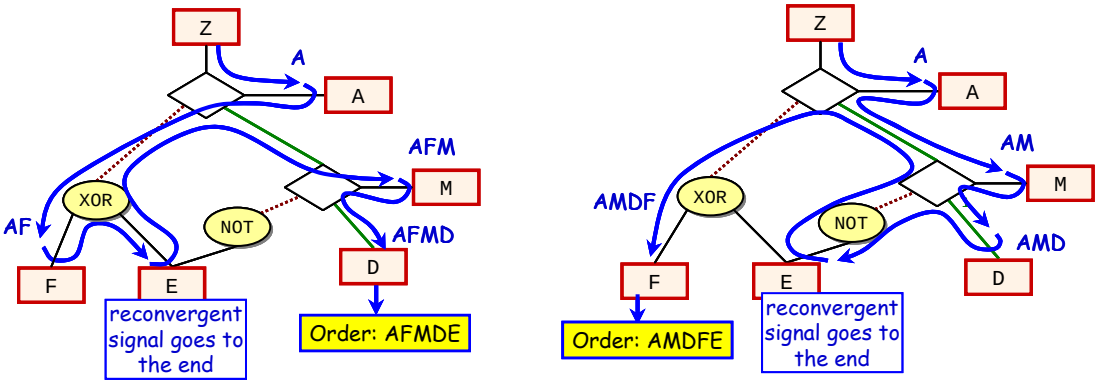


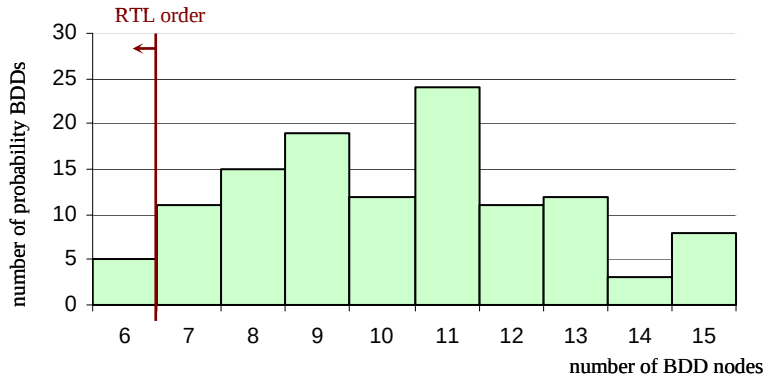
Figure E2-81: Routes through the EHM to obtain the RTL variable order of example 3

Some of the variable orderings have been included in table E2-9. In order not to have a large table, not all the orderings have been included. The RTL orderings have been highlighted in boldface type. The mean, the median, the first quartile and the first decile have been included at the bottom of the table. Note that the two RTL orderings are below the first decile.

Order	Nodes			Paths		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMDE *- AFMED -AMFDE - AMFED*	6	27	15	8	62	24
AMDFE	6	29	16	8	64	24
FAMDE - FAMED - MAFDE - MAFED	7	32	19	12	144	54
AMDEF	7	37	19	8	64	24
AMEDF - AMEFD	7	39	19	8	64	24
MADEF	7	34	20	12	144	44
ADMFE	7	36	21	10	100	37
ADFME - AFDME	7	37	22	10	100	37
MAEDF - MAEFD	8	44	23	12	144	44
AFEDM	9	47	27	10	100	43
EMADF - EMAFD	10	60	40	14	196	83
EDAFM - EDAMF	11	51	32	14	196	69
FMDAE - MDFAE - MFDAE	12	75	44	18	324	109
DFEMA - EFDMA - FDEMA - FEDMA	13	93	53	18	324	116
DFMAE - FDMAE	14	95	59	24	576	188
DFMEA - FDMEA	15	105	67	24	576	196
[...]						
Mean	10,27	60,13	36,05	15,23	250,7	89,65
Median	10	51,5	32	16	256	90
1st quartile	8	46	27	11,5	133	45
1st decile	7	37	19,9	10	100	42,4

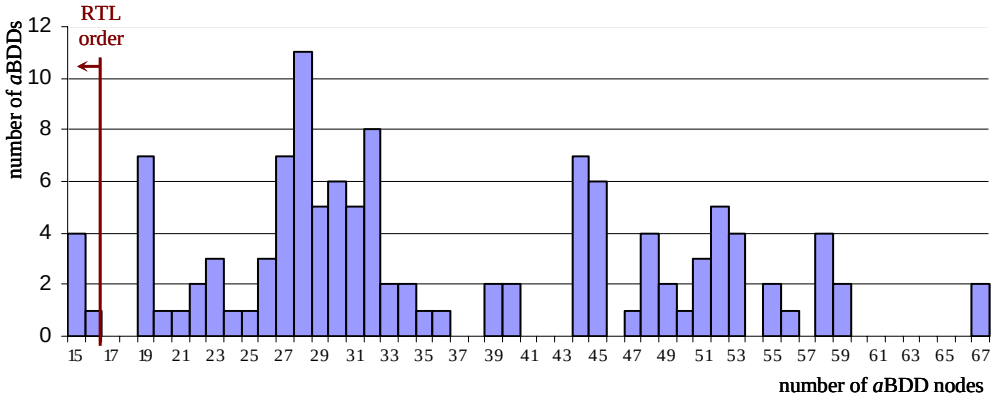
Table E2-9: BDD size of example 3 depending on the variable order

Because table E2-9 is not complete, the histogram of the 120 BDDs has been drawn in graphic E2-1. The X-axis indicates the number of nodes of the probability BDD. The height of each bar specifies how many probability BDDs have the number of nodes the X-value indicates. All the BDDs obtained by the RTL ordering are in the first bar.



Graphic E2-1: Distribution of the probability BDDs of example 3 depending on the variable order

Graphic E2-2 shows the distribution of the *a*BDDs depending on the variable order. As it can be seen, the RTL orderings are in the first two bars (15 and 16 nodes).



Graphic E2-2: Distribution of the aBDDs of example 3 depending on the variable order

E2.6.1.4 Example 4

This circuit has more than one reconvergent signal. Figure E2-82 shows the circuit and three of the 24 ordering possibilities. The circuit corresponds to the reconvergence area of the circuit of figure E2-19.

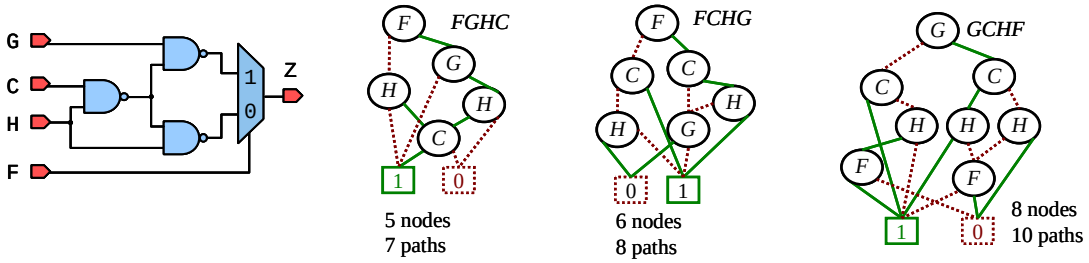


Figure E2-82: Some probability BDDs of example 4 for different variable orderings

The RTL ordering would take the signal of the conditional statement first (*F*), and the reconvergence signals would be at the end (*H* and *C*). In the middle would be signal *G*. Therefore, there are two RTL orderings depending on which reconvergent signal would be last: *FGHC* and *FGCH*.

The distribution of the BDD sizes is included in table E2-10. The distribution is more compact than the previous examples. The RTL orderings are within the four smaller BDD sizes. As usual, they have been highlighted in boldface type.

Order	Nodes			Paths		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
FGHC *- HCFG*	5	25	16	7	49	24
FGCH *- HCGF*	6	28	17	7	49	22
CHFG - GFHC	6	30	20	10	100	44
CFHG-FCHG-FHCG-FHGC *GHFC-HFCG-HFGC-HGFC*	6	34	22	8	64	31
CHGF-GFCH	7	33	21	10	100	40
CFGH-FCGH-GHCF-HGCF	7	37	23	8	64	31
CGFH-CGHF-GCFH-GCHF	8	45	28	10	100	44
Media	6,5	34,7	22	8,5	73.5	33,7
Mediana	6	34	22	8	64	31

Table E2-10: BDD size of example 4 depending on the variable order

E2.6.1.5 Example 5

Figure E2-83 shows the circuit of example 5. The circuit has a reconvergent signal affecting to a multiplexer alternative and affecting directly to the circuit output. It is a similar circuit to the circuit of example 3 but signal *D* is reconvergent and it directly affects to the output.

The two BDDs shown in figure E2-83 are the smallest probability BDDs, which are the unique BDDs with 6 nodes. These four orderings correspond to the RTL orderings: *DAFME*, *DAMFE*, *AFMED*, *AMFED*.

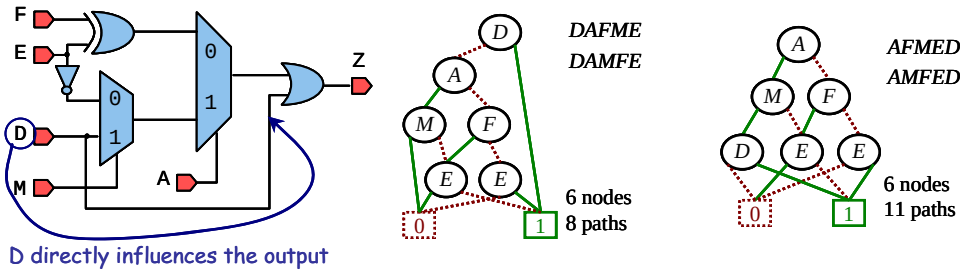


Figure E2-83: The smallest probability BDDs of example 5

The routes of the orderings *DAFME* and *DAMFE* are shown in figure E2-84. From signal *Z* the OR gate is reached, from which any of the two branches could be taken. Signal *D* can be taken because, even though it is a reconvergent signal, its branch is much simpler than the other (rule 9). Afterwards, the outer conditional statement is reached. From this point, the routes are similar to the routes of figure E2-81, but signal *D* is not added because it has been already included in ordering.

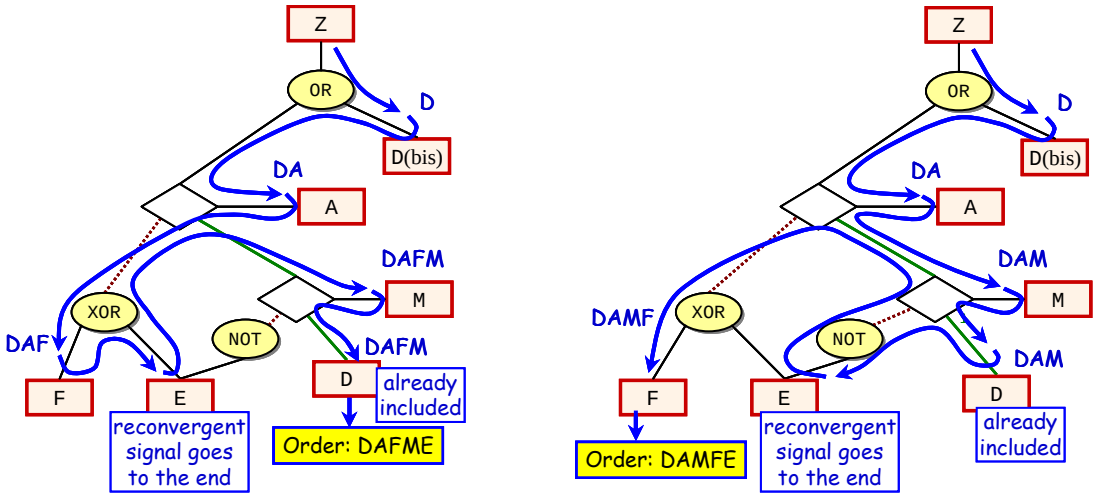


Figure E2-84: Routes (I) through the EHM to obtain the RTL variable order of example 5

The routes of the other two orderings (*AFMED* and *AMFED*) are shown in figure E2-85. In these orderings, once the OR gate is reached, instead of taking the branch of signal *D*, the branch of the multiplexer is taken (the rhombus) due to rule 7, as it recommends taking the more complex branch.

Once the multiplexer is reached, signal *A* is added (rule 1). Then, either of the branches could be taken: branch zero (left route of figure E2-85) or branch one (right route). In both routes, signal *E* is added when going out from its reconvergence area. Therefore, signal *E* precedes signal *D* because the reconvergence area of signal *D* covers the reconvergence area of signal *E*.

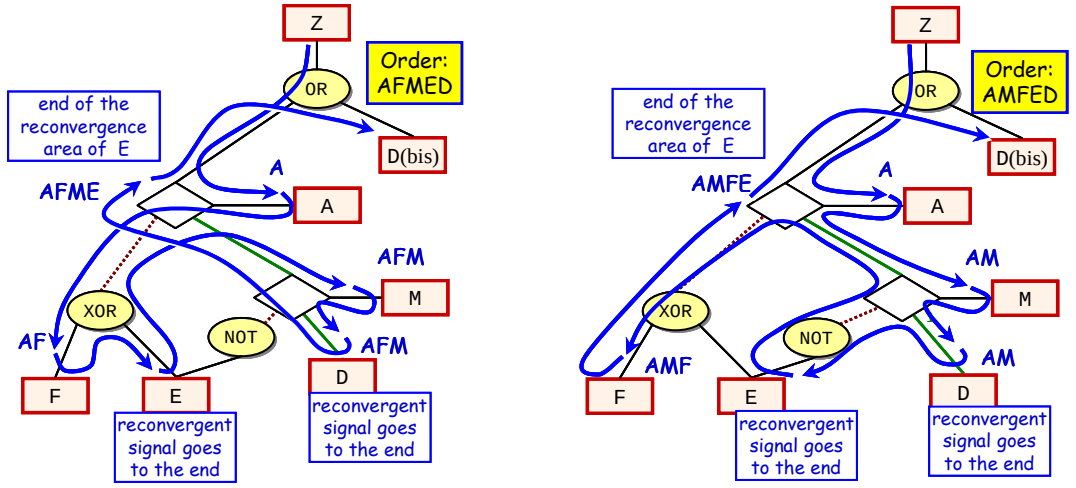


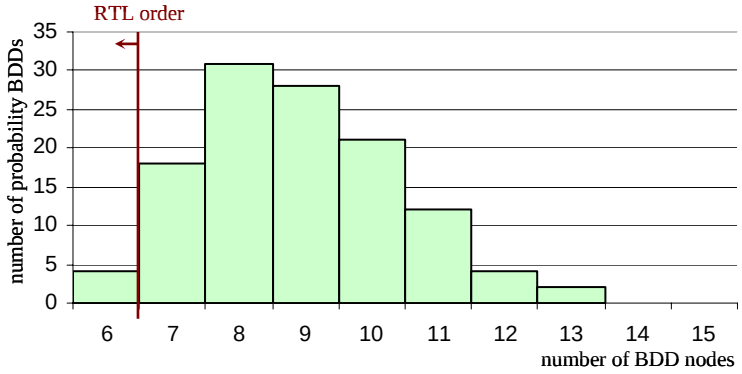
Figure E2-85: Routes (II) through the EHM to obtain the RTL variable order of example 5

Table E2-11 shows some of the 120 orderings. The first two rows are the only cases where probability BDDs have 6 nodes and these BDDs correspond to the RTL orderings, which have been highlighted in boldface type. At the end of the table, the mean, median, first quartile and first decile have been included.

Order	Nodes			Paths		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
DAFME - DAMFE	6	29	17	8	64	25
AFMED - AMFED	6	28	18	11	121	44
DAMEF	7	37	20	8	64	25
ADFME - ADMFE	7	34	21	9	81	34
AFMDE - AMFDE	8	41	26	11	121	54
AEMDF - EAMDF	9	53	32	11	121	58
EFMDA - FEMDA	10	61	38	14	196	90
FMADE - MFADE	11	63	43	19	361	153
FMDAE - MFDAE	12	84	56	16	256	109
FMDEA - MFDEA	13	93	61	16	256	111
[...]						
Mean	8,88	51,3	32,95	12,82	173,8	71,23
Median	9	48,5	32	12,5	156,5	62
1st quartile	8	42	26	11	121	48
1st decile	7	37	22	9	81	38

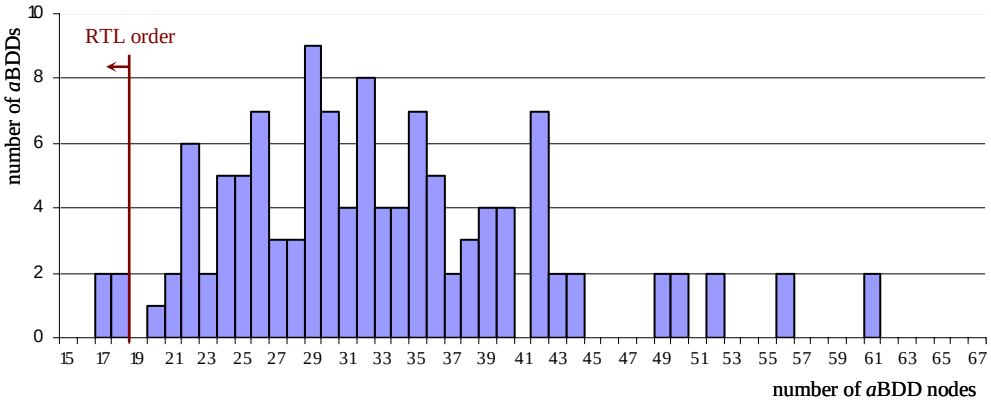
Table E2-11: BDD size of example 5 depending on the variable order

Graphic E2-3 shows the probability BDD size distribution of all variable orderings. Note that this histogram is more compact than the histogram of example 3 (figure 3-1). One reason could be that the more reconvergences a circuit has the more compact the BDD size distribution is.



Graphic E2-3: Distribution of the probability BDDs of example 5 depending on the variable order

The distribution of the *a*BDD sizes is shown in graphic E2-4. Note the situation of the RTL orderings in the histogram. Similarly to the probability BDD histogram, this histogram is more compact than the one of example 3 (graphic E2-3).



Graphic E2-4: Distribution of the aBDDs of example 5 depending on the variable order

The BDD size of the proposed ordering (17 or 18 nodes) is significantly lower than the BDD sized obtained in the worst case for TFBDDs (93 nodes) or the *a*BDDs (61 nodes). The

differences are still important compared with the median using TFBDDs (48-49 nodes) or *a*BDDs (32 nodes).

E2.6.1.6 Example 6

This circuit is a larger variation of example 5. As shown in figure E2-86, the circuit has a new input (*G*). Therefore, there are 720 possible orderings.

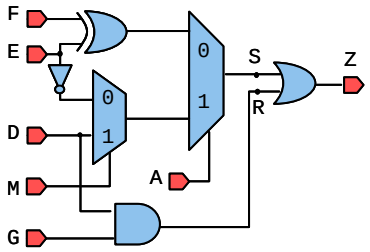


Figure E2-86: Circuit of example 6

There are 8 possible RTL orderings. Four of them result from taking the routes of figure E2-87. These routes are similar to the routes of example 5 but, now, there are two signals (*D* and *G*) in the first branch taken. Therefore, there are two possible orderings for each way: *GDAMFE*, *DGAMFE*, *GDAFME* and *DGAFME*. Because signal *R* is much simpler than signal *S*, the reconvergent signal *D* is not left to the end of the ordering; instead, it is added when it appears (rule 9).

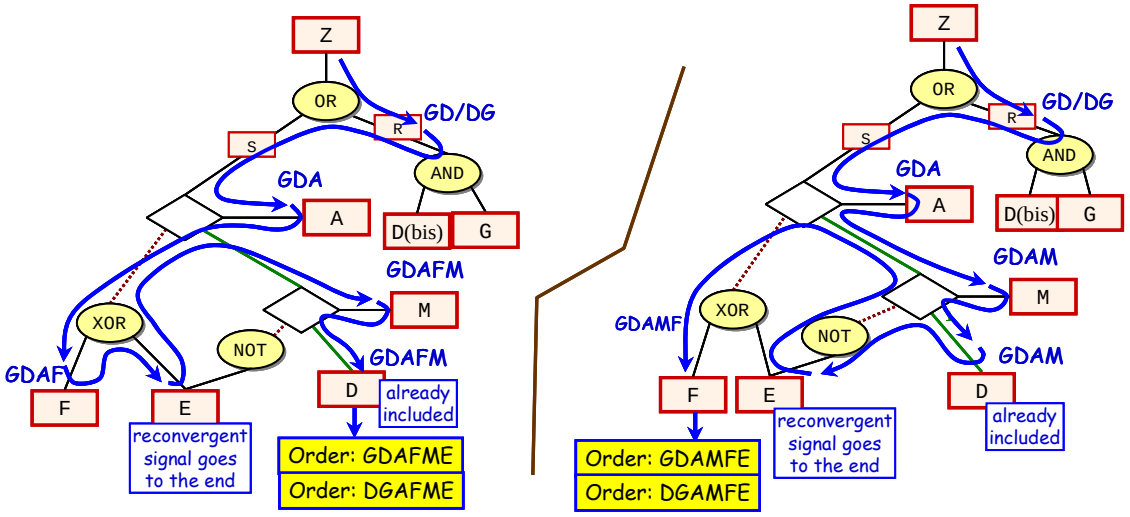


Figure E2-87: Routes through the EHM to obtain the RTL variable order of example 6

The other alternative is to first take the path going to the conditional statement (signal *S*). The orderings resulting from this option are: *AMFEGD*, *AMFEDG*, *AFMEGD* and *AFMEDG*. Note that signal *E* has been left until its reconvergence region has finished. The last signal is *D* because its reconvergence region covers the reconvergence region of signal *E*. The orderings *AMFEDG* and *AFMEDG* are obtained by rule 8, by which signal *D* is set when the way through signal *S* has finished. This route has not been drawn because is similar to the route of figure E2-85 (example 5).

The orderings that produce the smallest BDDs have been included in table E2-12. This table also includes the orderings producing the two largest BDDs and the mean, median, the first quartile and the first decile.

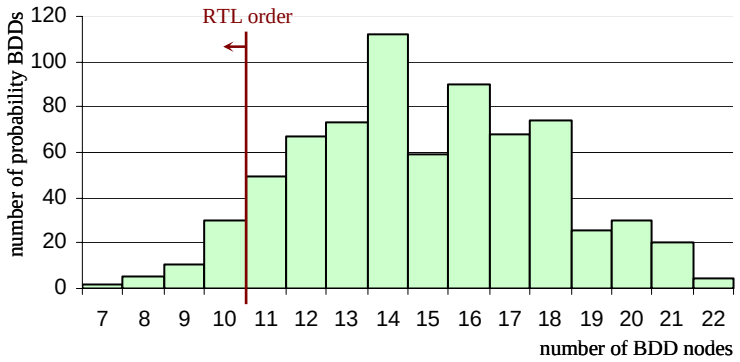
Order	Nodes			Paths		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMEGD - AMFEGD	7	37	23	14	196	74
FAMEGD	8	42	27	20	400	138
MAFEGD	8	42	27	22	484	158
AFMEDG - AMFEDG	8	43	27	14	196	80
AMEFGD	8	51	32	14	196	80
FAMEDG	9	48	31	20	400	152
MAFEDG	9	48	31	22	484	168
DGAMFE	9	52	32	15	225	76
DGAFME	9	55	34	15	225	76
AFEMGD	9	54	35	16	256	102
AEFMGD - EAFMGD	9	56	36	16	256	102
MAEFGD	9	56	36	22	484	170
AMEFDG	9	57	36	14	196	86
AEMFGD - EAMFGD	9	60	38	16	256	102
DGAMEF	10	60	35	15	225	76
GDAMFE	10	57	36	22	484	144
GDAFME	10	60	38	22	484	144
[...]						
FGMEDA - FMGEDA - GFMEDA - GMFEDA - MFGEDA - MGFEDA	21	245	152	37	1369	534
EGFDAM - FGEDAM - GEFDAM - GFEDAM	22	206	131	31	961	423
Mean	14,9	124,8	80,45	24,48	636,2	249,1
Median	15	121	76	24	576	229
1st quartile	13	89	57	19	361	148
1st decile	11	69	45	17	289	115,9

Table E2-12: BDD size of example 6 depending on the variable order

The RTL orderings produce small BDDs. The BDDs are smaller when the path of signal *S* (if-statement) is taken first and afterwards, the path of signal *R* (AND gate). Generally, it is better to choose the most complex path (or deepest path). In this case, the optimum RTL orderings are *AFMEGD* and *AMFEGD*, which have the reconvergent signal *D* at the end. Compare the size difference between this case (BDD: 7 nodes, *a*BDD: 23 nodes) and the worst case (BDD: 22 nodes, *a*BDD: 152 nodes). That is, more than 3 times the number of nodes for the BDD and more than 6 times the number of nodes for the *a*BDD.

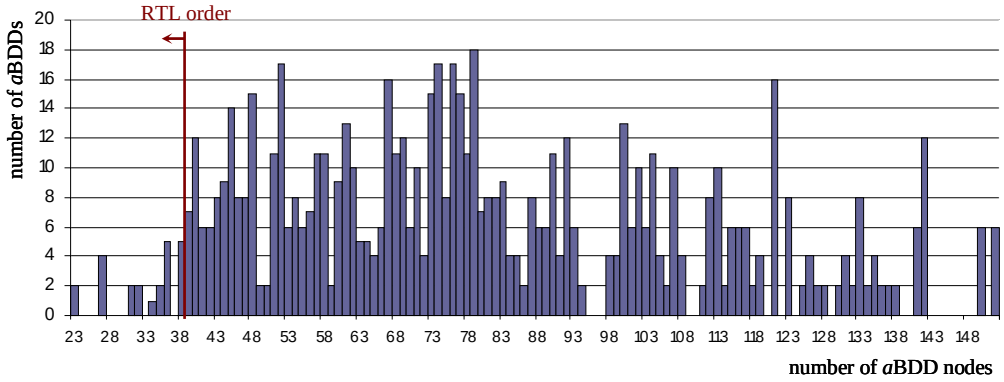
Even the worst RTL ordering is better than the median: 10 BDD nodes to 15 BDD nodes and 38 *a*BDD nodes to 76 *a*BDD nodes.

The distribution of the probability BDDs has been drawn on graphic E2-5.



Graphic E2-5: Distribution of the probability BDDs of example 6 depending on the variable order

The histogram of the *a*BDDs has been drawn in graphic E2-6. As it is usual for *a*BDDs, the histogram is more distributed and, as a result, the differences between the edges of the histograms are larger. Note where the RTL orderings (≤ 38) are located in the histogram.



Graphic E2-6: Distribution of the aBDDs of example 6 depending on the variable order

E2.6.1.7 Example 7

In order to prove the importance of the circuit structure to obtain the RTL ordering, this example has various circuits with the same structure but different logic gates. Therefore, all the circuits will have the same routes through the EHM and hence, they will all have the same RTL orderings. These four circuits have been drawn in figure E2-88.

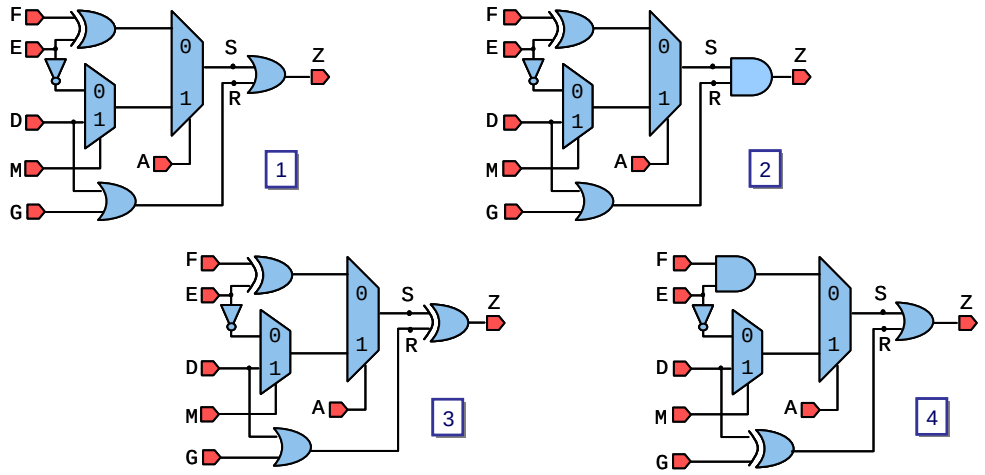


Figure E2-88: Circuits of example 7

The eight RTL orderings have been included in table E2-13. The table indicates the number of nodes of the BDD, the TFBDD and the *a*BDD of the RTL orderings. Besides, there also is a row that indicates the number of nodes of the ordering that produces the smallest probability BDD (*Min. BDD*). The next row indicates the number of nodes of the ordering that produces the smallest *a*BDD (*Min. aBDD*). These two rows may correspond to the same ordering. Then, two rows indicate the number of nodes of the orderings that produce the largest probability BDD and *a*BDD (*Max. BDD* and *Max. aBDD*). Afterwards, there are four rows that indicate the mean, median, first quartile and first decile of the 720 orderings. The penultimate row shows the node reduction between the worst case (*% red. max*) and the most favorable RTL ordering. The last row shows the node reduction between the median (*% red. max*) and the most favorable RTL ordering.

Order	BDD nodes of circuit 7.1			BDD nodes of circuit 7.2			BDD nodes of circuit 7.3			BDD nodes of circuit 7.4		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMEGD	7	32	21	7	37	23	9	44	28	9	47	30
AMFEGD	7	32	21	7	37	23	9	44	28	9	47	30
AFMEDG	7	32	21	8	43	27	9	44	28	9	47	30
AMFEDG	7	32	21	8	43	27	9	44	28	9	47	30
DGAMFE	7	33	20	9	52	32	10	57	36	10	57	37
DGAFME	7	33	20	9	55	34	10	60	38	10	60	39
GDAMFE	7	33	20	10	57	36	10	57	36	10	57	37
GDAFME	7	33	20	10	60	38	10	60	38	10	60	39
[...]												
Min. BDD	7	32	21	7	37	23	9	44	28	9	47	30
Min. aBDD	7	33	20	7	37	23	9	44	28	9	47	30
Max. BDD	17	133	91	22	206	131	23	249	158	22	244	159
Max. aBDD	17	133	91	21	245	152	23	249	158	22	244	159
Mean	10,9	68,2	45,3	14,9	124,8	80,2	16,6	141,0	90,6	15,2	126,0	82,9
Median	11	65	43	15	121	76	17	137	86,5	15	212	78
1st quartile	9,75	54	36	13	89	57	15	103,5	65	13	101	67
1st decile	9	47	31	11	69	45	13	80	52	12	78,9	52
% red. max.	58,8	75,9	78,0	68,2	84,9	84,9	60,9	82,3	82,3	59,1	80,7	81,1
% red. med.	36,4	50,8	53,5	53,3	69,4	69,7	47,1	67,9	67,6	40,0	77,8	61,5

Table E2-13: BDD size of example 7 depending on the RTL variable order

Note that better orderings are obtained when the path of signal *S* is taken first taking. Therefore, as it has been said, if the path of the most complex signal is taken first, it produces better orderings (rule 6).

The worst RTL cases always are better than the median; in fact, all of them are in the first decile.

E2.6.1.8 Example 8

This example is a more complex circuit. This circuit, shown in figure E2-89, has seven inputs; thus, there are 5040 ordering possibilities.

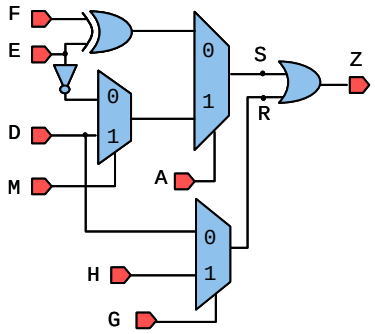


Figure E2-89: Circuits of example 8

There are eight RTL orderings, which have been included in table E2-14. The RTL orderings are *AFMEDGH*, *AMFEDGH*, *AFMEGHD*, *AMFEGHD*, *GHDAMFE*, *GHDAFME*, *GDHAMFE* and *GDHAFME*.

Orderings *AFMEDGH*, *AMFEDGH*, *AFMEGHD* and *AMFEGHD* are obtained following rule 7. Taking first signal *S* because it is more complex than signal *R*. When the way through signal *S* has finished, signal *D* can be added to the ordering by rule 8, resulting in the orderings *AFMEDGH* and *AMFEDGH*. If signal *D* is not added because of its reconvergences, it will be set at the end: *AFMEGHD* and *AMFEGHD*.

Orderings *GHDAMFE*, *GHDAFME*, *GDHAMFE* and *GDHAFME* are obtained using rule 9. Taking first the simplest signal *R* and not leaving the reconvergent signal *D* to the end. Note that these orderings are less favorable.

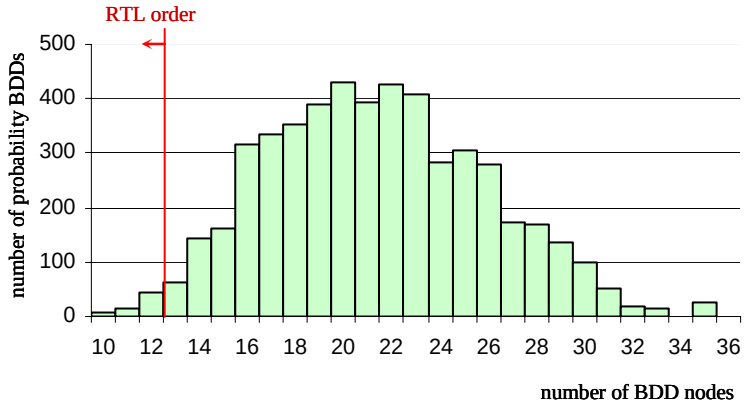
Table E2-14 also includes the worst orderings (*HMFDEGA*, *HFMDEGA* y *MHFDEGA*). The table includes the mean, median, first quartile, first decile and 5th percentile. Note that all the RTL orderings are in the 5th percentile.

The last two rows indicate the percentage of the reduction achieved by the first row in respect to the worst case (*% red. max*) and the median (*% red. median*).

Order	Nodes			Paths		
	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
AFMEDGH - AMFEDGH	10	55	36	25	625	219
AFMEGHD - AMFEGHD	10	59	38	20	400	152
GHDAMFE	11	64	41	23	529	167
GHDAFME	11	67	43	23	529	167
GDHAMFE	12	73	47	24	576	191
GDHAFME	12	76	49	24	576	191
[...]						
HMFDEGA - HFMDEGA - MHFDEGA	35	587	374	52	2704	1176
Mean	21,4	235,4	153,1	37,9	1523,8	617,3
Median	21	221	145,5	38	1444	578
1st quartile	18	162	106	30	900	395
1st decile	16	124	80,9	26	676	293
5th percentile	14	104	67	24	576	240
% red. max.	71,4	90,6	90,4	51,9	76,9	81,4
% red. median	52,4	75,1	75,3	34,2	56,7	62,1

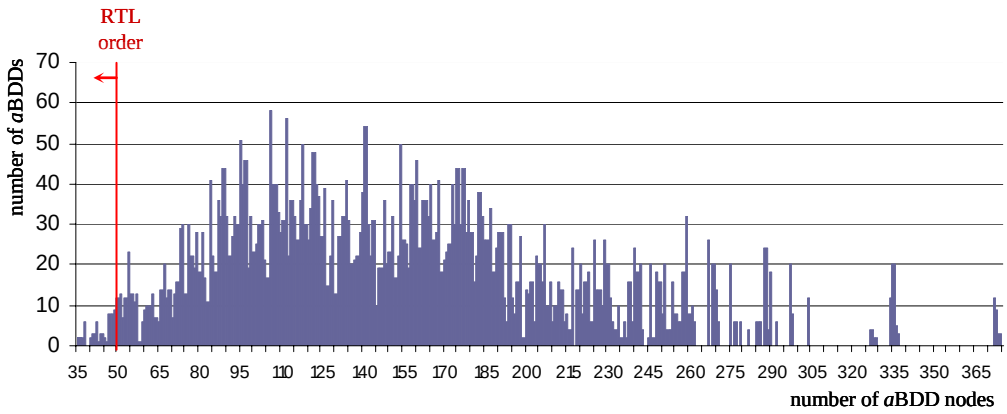
Table E2-14: BDD size of example 8 depending on the RTL variable order

The histogram of the number of nodes of the probability BDDs has been drawn in graphic E2-7.



Graphic E2-7: Distribution of the probability BDDs of example 8 depending on the variable order

The histogram of the number of nodes of the *a*BDDs has been drawn in graphic E2-8.



Graphic E2-8: Distribution of the aBDDs of example 8 depending on the variable order

E2.6.2 Conclusions about the RTL Order

The examples support the assumption that the order provided by the RTL analysis is suitable. Other tests have been carried out with circuits of similar size and the results are analogous. These results have not been included because they do not offer anything new to the former examples.

The results showed that, in most cases, the RTL orderings were in the first decile and, in many cases, one of the RTL orderings was the optimum.

The changes of gates that have been carried out in the examples 6 and 7 showed that the circuit structure is important to determine the ordering. However the logic function also affects because the more complex a logic function is, the sooner it has to be traversed.

The proposed rules are open and may provide more than one ordering. Generally, any of them is valid, but if only one ordering should be taken, the rule would be to take first the most complex alternative (rule 6) and to leave the reconvergent signal to the end (rule 5). Nevertheless, the proposal of more than one ordering may be considered as an advantage because, instead of taking a risk with only one ordering, the proposal allows the selection of the best order within a reduced set. For example, eight RTL orderings out of 5040 possible orderings were proposed in circuit 8.

Last, as the number of inputs grows, BDD size differences between different orderings also increase. Therefore, it is crucial to have an appropriate ordering method. The examples have shown that the RTL ordering keeps the BDD sizes in the lower zone of the histogram. Moreover, when the number of inputs grows, the RTL orderings are in lower percentiles.

As a conclusion, although the method is not definitive and has no theoretical base, the results lend support the assumption that it **is a valid and immediate method to obtain an adequate BDD variable ordering**.

E2.7 Probability and activity propagation

The analysis process explained up to this point in the Thesis is:

- The extended hardware model of the circuit is built (EHM, §E2.3.2)
- The EHM is traversed from signals with lower combinational depth to those with higher combinational depth (§E2.3.3)
- During the EHM traversal, the circuit is partitioned in reconvergent regions. If regions are still large, the circuit may be also partitioned in disjoint regions (§E2.4)
- The probability and activity BDDs are built for each region (§E2.5). Activity BDDs are built with a new BDD representation. BDDs are built following an ordering based on the RTL structure

Therefore, up to this point, the probability and activity BDDs for any signal are available. Since the analysis is performed starting from the lowest combinational depth signals, the numeric values of the dependencies of the signals being analyzed are already calculated. Then, the numeric values of the probability and activity of the signal can be computed. Annex AI.2 (in Spanish) briefly explains the algorithm used to calculate the probability and activity numeric values from the BDDs.

Once the probability and activity numeric values have been calculated, they are written down in the EHM in order that these values can be used to compute the activity of the next signals.

Before the probability and activity values are propagated through the circuit, it is necessary to assign the probability and activity values of the inputs.

The probability and activity propagation and its annotation in the EHM results in a probability and activity map as the shown in figure E2-90 for a sample circuit.

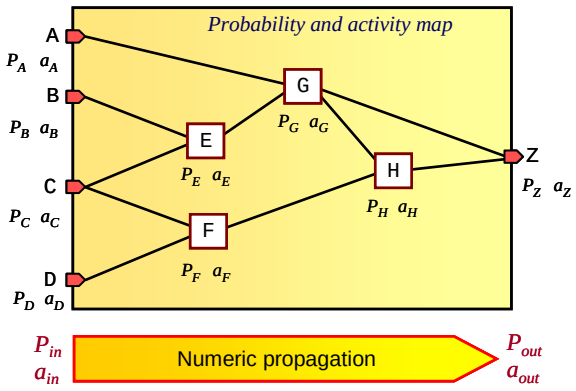


Figure E2-90: Example of a probability and activity map

Once the analysis have been carried out and the activity map has been obtained, it is not necessary to repeat the whole process if initial conditions change. All the information and processing of the previous sections is kept in the EHM; therefore, only the numeric values have to be propagated through the same structure and BDDs. Consequently, this last step is fast, what provides with an efficient way to know the circuit activity in different input conditions. This is a clear advantage of probabilistic methods compared with simulative ones, which require a complete new simulation each time the input conditions change.

E2.8 Conclusions

This chapter has explained the activity estimation method proposed in this Thesis. The proposal consists in raising the abstraction level of the probabilistic analysis from the logic level to RTL. The method is the first approach at this level; therefore, it does not intend to be complete. However, the advantages that are arising from the method encourage further investigation.

Raising the abstraction level has many implications on the estimation method. Therefore, the proposal has faced new difficulties, on the contrary, the proposal has benefited from the advantages of the analysis in a higher abstraction level. Consequently, the general proposal of this Thesis consists of some particular proposals that overcome the difficulties and exploit the new abstraction level. All these individual proposals contribute to the main objective of the Thesis: **to raise the abstraction level of the switching activity estimation** from the gate level to RTL.

The variety of the particular proposals may complicate the comprehension of the method. Thus, in order to improve the readability and provide less weighty information, some sections have not been included in the chapter but in annex II (in Spanish). Although these additional sections are useful to study in depth the proposal, they are not necessary to comprehend it.

Following, the proposals and advantages of the method will be summarized:

The **fundamental proposal of this Thesis** (to raise the abstraction level of the activity estimation) **permits to perform earlier estimations**. The estimation results help to make correct design decisions. Since estimation results are now available in a previous design phase, the method contributes to **reduce the design iterations**.

In addition, raising the abstraction level to RTL provides practical advantages to the analysis.

- There is **more high-level information** available at RTL. This information is provided by the logical and mathematical operators, the RTL structures such as multiplexers and the signal grouping in vectors
- The analysis is simpler because RTL designs are not as detailed as gate level designs

However, the higher simplicity of RTL designs also entails inconveniences because some of the gate level information is necessary for the analysis. For example, RTL does not have a detailed structure of signal dependences and influences as the gate level has with netlists. As a consequence, this Thesis proposes the creation of a **hardware model** (EHM) to provide a structure in which the analysis can be performed (§E2.3). The advantages of the hardware model are:

- It provides a **structure** of dependences and influences similar to gate level netlists
- It **preserves the RTL information** in order to be used for the probabilistic analysis
- It **maintains the simplicity**, avoiding the inconveniences of heavily detailed descriptions such as the gate level are

For large circuits, it is necessary to partition the circuit in order to limit the computational and memory resources. However, partitions must consider the circuit structural dependences so as to avoid errors. Reconvergent partitions provide exact probability and activity values. Partitioning at RTL instead of at gate level has the following advantages:

- The **reconvergence is kept**. That is, reconvergences may increase after the logic synthesis and thus, the partitions' size may increase too
- The **partition process is simpler** because the RTL description is less detailed than the gate level description

Nevertheless, in many cases, partitions may still be too large. In consequence, this Thesis proposes **the disjoint region partition at RTL** (§E2.4) due to their **smaller** sizes. Although disjoint partition has been proposed at gate level [2], the work was only related to signal probability and not signal activity. Besides, contrary to what happens at gate level, identifying disjoint regions at RTL is **simple** (§E2.4.1). Therefore, this Thesis proposes to identify the disjoint regions at RTL.

Signal probability calculation using disjoint regions is exact; however, disjoint region partitions introduce approximations that produce some error. The effect of these approximations in the activity values has been evaluated in this Thesis (§E2.4.3). As a result, the Thesis proposes the conditions under which disjoint partitions provoke **an acceptable error**.

This Thesis uses BDDs to calculate signal probabilities and activities. Both activity BDDs and equations are more complex than those of probability. Consequently, this Thesis proposes a **new representation of activity BDDs** (§E2.5.2). Moreover, the Thesis includes the theoretical formulation of its representation (§E2.5.2.4). This proposal achieves:

- A node reduction from 25% to 50% with no cost in exactitude.
- The theoretical formulation permits the creation of an activity operator that has been integrated in the BDD library package (BuDDy [64]) that has been used to manipulate BDDs.

The BDD variable order has an enormous influence on the BDD size. The ordering algorithms are complex due to the fact that for a BDD of n variables, there are $n!$ ordering possibilities. In consequence, this Thesis **proposes a variable ordering based on the RTL structure of the circuit** (§E2.6). An empirical study of the ordering considering the RTL structure has been carried out in this Thesis. As a result, **a set of rules have been proposed so that BDDs can be efficiently ordered in a systematic way**. The results of the ordering method are:

- Contrary to what happens at gate level, **the proposed ordering method is simple to carry out at RTL**. Thus, complex algorithm methods are avoided
- The proposal of rules permit the process **automation**
- The **experimental results support the method validity**. From all the ordering possibilities, in most cases, the RTL orderings are in the first decile.

Besides, this Thesis proposes the usage of the **vector index information available at RTL** in order to achieve a better BDD ordering. Knowing the **vector indexes helps to attain an optimum ordering depending on the operation** the signals are involved in. The information

related to the vector indexes is not available at gate level. This topic will be extended in chapter E3.

Finally, propagation of the probability and activity values is performed through the EHM. Since the proposed estimation method is static, creating the probabilistic model entails a computation intensive process. However, once the probabilistic model has been created, the probability and activity values can be effortless recomputed when input conditions change.

All the proposals have been **implemented in a CAD tool**, which demonstrates the practical feasibility of the proposed method. Besides, it permits several test to be carried out over medium size circuits. Without the CAD tool, these experiments would not have been possible.

To conclude, the proposed activity estimation method may constitute the base for probabilistic estimation at RTL. The proposal is constituted by various individual contributions that make possible the estimation at RTL and benefit from the advantages the new abstraction level offers. The next chapter will carry out experiments in order to confirm the feasibility of the proposal.

E3. EXPERIMENTAL RESULTS

This chapter intends to demonstrate the experimental validity of the Thesis' proposal. The fundamental proposal of this Thesis is to raise the switching activity probabilistic estimation from the gate level to RTL. The theoretical advantages of this proposal have been expounded in the previous chapter. The objective of this chapter is to demonstrate that these theoretical advantages are also proved in practical experiments.

The previous chapter explained that, because of the magnitude of the proposal, this Thesis contributes with diverse solutions that resolve particular questions which constitute the main proposal. These particular proposals can be summarized in.

1. Elaboration of an extended hardware model (EHM §E2.3.2)
2. Circuit partition in disjoint regions (§E2.4)
3. New representation of the activity BDDs (*a*BDD §E2.5.2)
4. Proposal of a BDD ordering based on the RTL structure (§E2.6)

In order to demonstrate the proposal, various circuits have been analyzed with the proposed method. The EHM has been elaborated for both the gate level and the RTL descriptions. For the RTL descriptions two variants of the EHM have been elaborated: one with no disjoint partition and the other with disjoint partition. For each of these three hardware models, the proposed activity BDD (*a*BDD) and the TFBDDs have been built to be compared with each other. Once both EHM and BDDs have been built, the probability and activity numerical values are propagated from inputs to outputs through the EHM. Last, the results obtained from each of the methods are compared.

With the intention of demonstrating the advantages of the RTL analysis and to prove the four particular proposals, this chapter analyzes:

a. The size variation of the BDDs obtained from the analysis at RTL with no disjoint partition against those obtained from the analysis at gate level. This experiment intends to prove the advantages of the RTL analysis against the gate level analysis and the advantages of the RTL ordering (proposal 4)
b. The size difference of the BDDs obtained from the analysis at RTL with and without disjoint partition. The experiment intention is to demonstrate the disjoint partition advantages (proposal 2). Only at RTL the disjoint partition can be carried out in a simple way; thus, it is an advantage of the RTL analysis against the gate level analysis. Because disjoint partitions do not produce exact results, the error is also analyzed
c. The size variation of the activity BDDs using *a*BDDs instead of TFBDDs (proposal 3)

The usage of EHM (proposal 1) is an essential step for the analysis at RTL because it provides with the necessary structure for the probability and activity propagation (§E2.3). Therefore, the validation of this proposal is demonstrated as a necessary mean to carry out the activity analysis. Besides, without the EHM it would not be easy to identify the disjoint regions of a circuit. Therefore, EHM also constitutes an intermediate step necessary for carrying out proposal 2. Last, the EHM elaboration is a one time process, since once it has been built, it is not necessary to repeat the analysis when input conditions changes. This is an important advantage of probabilistic models.

In consequence, if the analyses applied to the test circuit are favorable for the RTL approach, the theoretical demonstrations of the previous chapter will be proved. As a result, it could be assumed that this Thesis' proposals are valid.

This chapter is organized as follows. In section E3.1, a brief description of the circuits used in the experiments is given. Then, in section E3.2, the different probabilistic models are compared for each circuit. Afterwards, section E3.3 analyses the error caused by the disjoint region partition. In section E3.4, instead of carrying out a circuit based analysis as in the previous sections, a global analysis considering all the circuits is performed for each of the individual

proposals. In section E3.5 a brief indication of the processing times is given. To end, in section E3.6 the conclusions of the experimental results are presented. Besides, in annex III (in Spanish), additional information and analyses have been included. This information has been separated in order to simplify the description of the results.

E3.1 Description of the circuits used in the experiments

In this section the VHDL circuits used in the experiments are described. The designs have been obtained from two sources. The first three designs are part of a set of RTL circuits proposed by Z. Navabi [144]. In this source, the designs are described in both RTL and gate level.

The rest of the circuits have been obtained from the ALU of the 8051 microcontroller provided by *Oregano Systems* [101]. These designs are generic; thus, the bus width of the operands can be modified. The gate level descriptions of these designs have been obtained through *Synopsys Design Compiler* [125]. In some cases, with the intention of analyzing the differences between different versions of the same circuit, some versions of the same gate level circuit have been obtained by changing the synthesis options.

Table E3-1 summarizes the circuits used. In order to perform a wider analysis, the bus width of some of the circuits has been modified, what also permits to study how the circuit size affect to the different analysis: gate level, RTL and disjoint-RTL.

no.	Circuit name	Bus width	Total gates	2-input gates	Inv.	Input ports	Output ports	Source	Description
1	max	4	26	22	4	8	4	[144]	It gives the biggest operand
		5	34	29	5	10	5		
		6	42	37	5	12	6		
		7	50	44	6	14	7		
		8	57	51	6	16	8		
		9	65	58	7	18	9		
		10	71	65	6	20	10		
		11	79	72	7	22	11		
		12	87	79	8	24	12		
		13	97	86	11	26	13		
		14	105	93	12	28	14		
		15	110	100	10	30	15		
		16	117	107	10	32	16		
		17	127	114	13	34	17		
		18	132	121	11	36	18		
		19	139	128	11	38	19		
		20	147	135	12	40	20		
2	comparador	4	30	25	5	11	3	[144]	Cascadable Comparator
		5	36	30	6	13	3		
		6	42	35	7	15	3		
		7	51	42	9	17	3		
		8	57	48	9	19	3		
		9	63	53	10	21	3		
		10	72	59	13	23	3		
		11	79	66	13	25	3		
		12	83	70	13	27	3		
		13	95	76	19	29	3		
		14	96	81	15	31	3		
		15	105	89	16	33	3		
		16	109	92	17	35	3		
		17	117	99	18	37	3		
		18	125	106	19	39	3		
		19	132	111	21	41	3		
		20	140	118	22	43	3		
		21	146	123	23	45	3		
		22	150	127	23	47	3		
3	alu_peq	4	510	371	139	11	7	[144]	4-bit ALU
4	alu8051_core	8	219	201	18	22	10	[101]	8051 ALU core
5	addsub	7	70	62	8	16	10	[101]	Adder/subtracter
		8	72	63	9	18	11		
		9	87	75	12	20	13		
		10	101	87	14	22	14		
		11	108	94	14	24	15		
		12	117	100	17	26	16		
		13	125	107	18	28	18		
		14	139	119	20	30	19		
		15	146	126	20	32	20		
6	alu8051_simp	7	383	348	35	23	10	[101]	Simplified 8051 ALU
		8	415	375	40	25	11		
		9	450	412	38	28	13		
		10	487	447	40	30	14		
		11	517	474	43	32	15		
		12	557	503	54	34	16		
7	alu8051	6	520	483	37	27	9	[101]	8051 ALU
		7	571	532	39	30	10		
		8	629	580	49	33	11		
		9	685	636	49	37	13		
		10	759	689	70	40	14		

Table E3-1: Summary of the circuits analyzed

The column "*Total gates*" of table E3-1 refers to synthesized circuits and it is an approximate value because it depends on the synthesizer and the synthesis options. The synthesis has been performed with only 2-input gates and inverters. Thus, 2 columns have been added to distinguish them: "*2-input gates*" and "*Inv.*".

Since this Thesis perform the analysis at RTL and the other proposals perform it at gate level, the results of the Thesis could not be compared with other works. The majority of the other works ([28], [119], [80], [7]) has used ISCAS'85 circuits [15]. These circuits are described at gate level; therefore, it is not possible to prove the RTL analysis advantages with these circuits. Hence, other circuits have had to be analyzed in order to demonstrate the advantages of RTL analysis.

As it can bee seen in table E3-1, all the circuits are arithmetical. This is due to the fact that, usually, control circuits are mainly sequential and their combinational parts are not too large. ISCAS'85 circuits are mainly arithmetical circuits too. ISCAS'85 benchmark is composed by ten combinational circuits, in which there are 4 ALU, a multiplier, an adder/comparator, an interrupt controller and 3 error detector circuits where two of them are the same circuit but described with different gates [53].

Besides, commonly, the combinational part of a control circuit is mainly constituted by multiplexers. As it has been explained in chapter E2, multiplexers benefits the RTL analysis, because they facilitate the RTL ordering (§E2.6) and permit the disjoint region partition (§E2.4).

No circuit from other sources has been taken due to the circuit adaptation effort needed for the analysis. The computer program that has been created to build the models and perform the estimations is a preliminary version, which only accepts a reduced set of VHDL (§E2.2). Therefore, each circuit has to be manually adapted. The adaptation process basically consists in the vector substitution into bit signals and the circuit flattening (that is, to eliminate the circuit hierarchy). These operations imply a considerable effort, especially for large circuits. For the case of the 8051 ALU, some parts of the design have been flattened and then, theses parts have been joined in a larger design. Since these parts perform different tasks, it could be considered that the results are independent. On the other hand, this incremental analysis allows the investigation of the implications of the increase of a circuit functionality and its size over the probability analysis.

In the following subsections, a brief description of each circuit is given.

E3.1.1 Circuit "max" (1)

This circuit has been obtained from Navabi [144]. It is a simple circuit that gives the biggest operand. Figure E3-1 shows a schematic view of the circuit and its VHDL code. The gate level description has 28 two-input gates (ADN & OR) and 11 inverters.

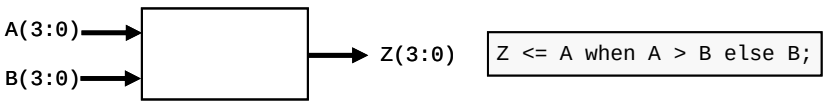


Figure E3-1: Inputs and VHDL code of circuit "max" (1)

E3.1.2 Circuit "comparador" (2)

This second example has also been obtained from Navabi [144]. It is a cascadable comparator. Figure E3-2 shows a schematic view of the circuit and its VHDL code. The gate level description has 29 two-input gates (AND & OR) and 17 inverters.

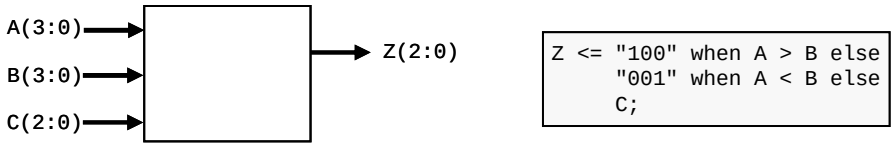


Figure E3-2: Inputs and VHDL code of circuit "comparador" (2)

E3.1.3 Circuit "alu_peq" (3)

The third circuit (Navabi [144]) is a simple ALU that has 8 operations. The operands (*A*, *B*) are 4 bits width and the operations are defined by the input *Code*. The operation result is given by output *Z* and the operation status is given by the signal *Flags*. The model has 371 two-input gates (AND & OR) and 139 inverters. The large number of gates is partly due to the fact that XOR gates have not been used.

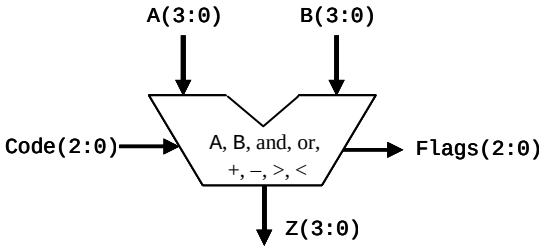


Figure E3-3: Schematic and VHDL code of circuit "alu_peq"(3)

E3.1.4 Circuit "alu_core" (4)

The fourth circuit is the core of the ALU of the microprocessor 8051 provided by *Oregano Systems* [101]. The circuit is simple because there only are logic operations in the core. Figure E3-4 shows the circuit inputs and outputs.

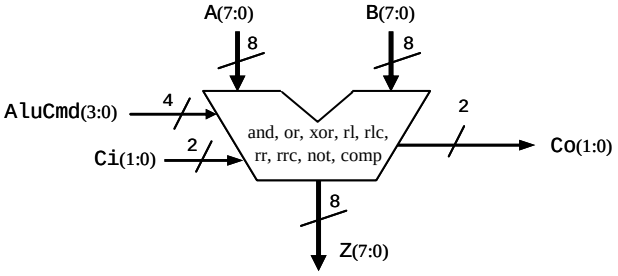


Figure E3-4: Schematic of circuit "alu_core" (4)

Although the operands bus width is generic, eight bits have been chosen for the experiments. This circuit has more input ports than the previous; however, due to the fact that the operations are at bit level and they are simple, this circuit has less number of gates (219).

Nine operations are included: AND, OR, XOR, rotation and shift to the right and left, inversion and comparison. The operations are selected by the input *AluCmd*. When the code of *AluCmd* does not match with any operations the outputs are zero.

E3.1.5 Circuit "addsub" (5)

The fifth circuit is the block adder-subtracter of the ALU of the microprocessor 8051 provided by *Oregano Systems* [101]. The block inputs and outputs are shown in figure E3-5. The bus width of the operands is defined by generic *n*.

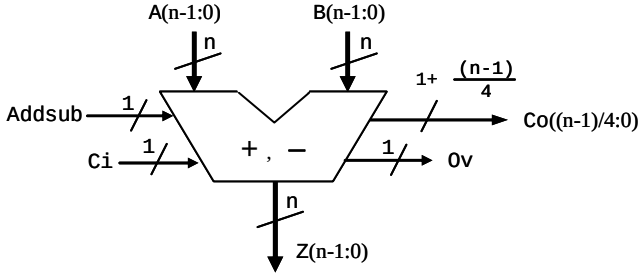


Figure E3-5: Schematic of circuit "addsub" (5)

The circuit inputs are:

- *Addsub*: indicates whether an addition or subtraction has to be performed. When *Addsub*='1', the addition is performed. When *Addsub*='0'; the subtraction is performed by calculating the two's complement of the subtrahend (*B*).
- Operand *A* is a term of the addition or the minuend of the subtraction. The bus width is *n*.
- Operand *B* is a term of the addition or the subtrahend of the subtraction. The bus width is *n*.
- *Ci* is the input carry

The circuit outputs are:

- *Z* is the result of the addition or the subtraction
- *Ov* indicates if there has been overflow
- *Co* is the carry out. It has 1+(*n*–1)/4 bits. The most significant bit is the carry out bit. The other bits correspond to nibble partial carry out.

E3.1.6 Circuit "alu8051_simp" (6)

This circuit joints the two previous circuits through multiplexers and some additional logic. It has the suffix *simp* to indicate that it has been "simplified" because it is not the complete 8051 ALU yet. The port diagram is shown in figure E3-6.

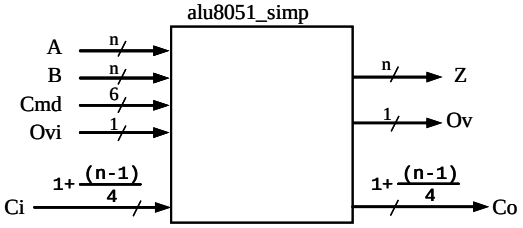


Figure E3-6: Schematic of circuit "alu8051_simp" (6)

Then, the internal circuit schematic is shown in figure E3-7, where the adder/subtracter and the ALU core can bee seen. The schematic has been particularized for 8 bits, but any bus width could be used instead.

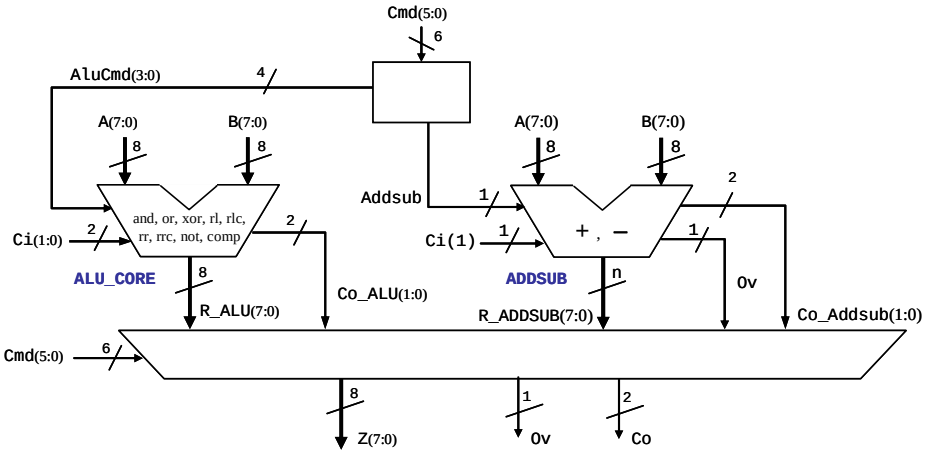


Figure E3-7: Internal schematic of circuit "alu8051_simp" (6)

E3.1.7 Circuit "alu8051" (7)

This circuit is the complete 8051 ALU, although its optional blocks are not included: the multiplier, divisor and decimal adjust. The difference with the simplified ALU is that now the operands are selected from the ROM, the RAM and the accumulator. Therefore, a multiplexer selects two operands from these three sources. The port diagram is shown in figure E3-8.

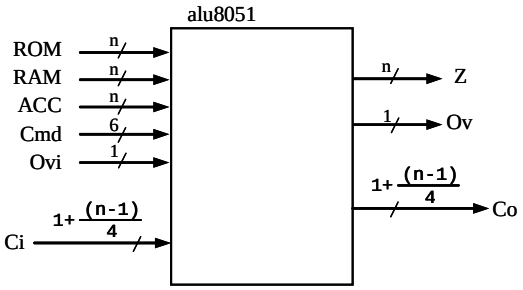


Figure E3-8: Schematic of circuit "alu8051" (7)

The internal schematic has been drawn in figure E3-9.

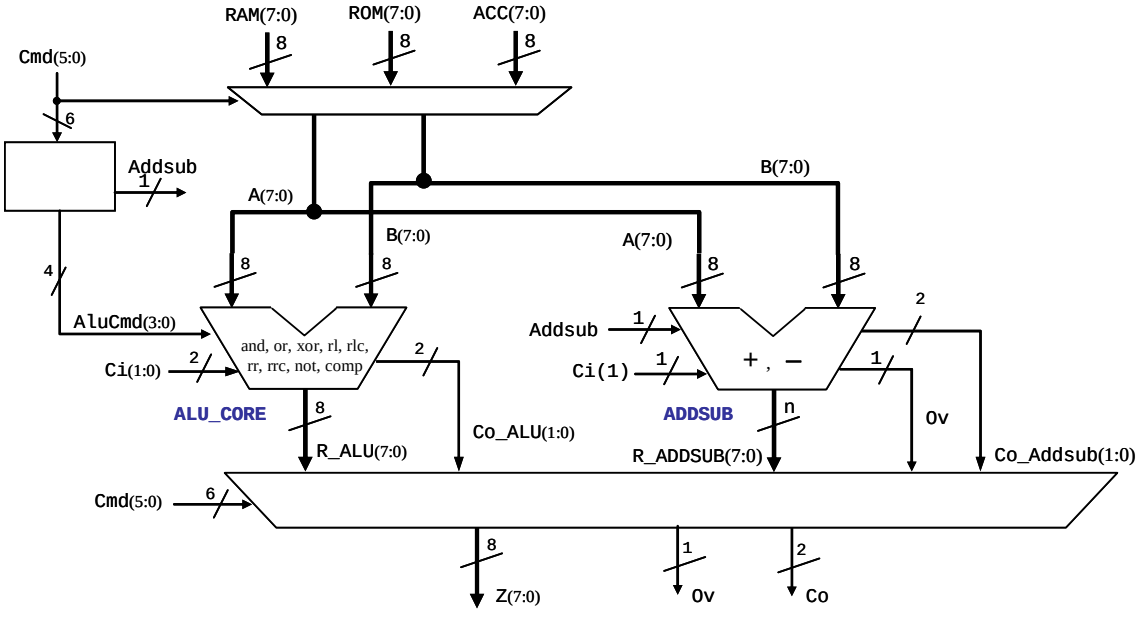


Figure E3-9: Internal schematic of circuit "alu8051" (7)

E3.2 Analysis of the models of each circuit

Since the computer program that has been made for this Thesis does not accept all the synthesizable VHDL set [60], the circuits have been adapted: flattening and vector substitution. Thus, the probability analyses have been done bit by bit, not considering vectors. Nevertheless, the vector indexes have been used to obtain adequate orderings.

For each circuit, the size of the resulting BDDs is compared between three different analyses:

- Gate level
- RTL without disjoint region partition (*RTL* or *exact-RTL*)
- RTL with disjoint region partition (*disjoint-RTL*)

The reconvergent partition is performed in all the analysis, but only in the last one, the disjoint partition is carried out. To simplify the nomenclature, these three analyses will be named as: *gate*, *RTL* (or *exact-RTL*) and *disjoint-RTL*.

The probability and activity results of the two first analyses are identical. Therefore, it makes sense to take the faster one or the one that consumes fewer resources. The third analysis (*disjoint-RTL*) is not exact; consequently, the trade-off between exactitude and resource consumption should be studied.

The models will be compared using tables that have two column groups (see table E3-2):

- The first column group has been called **Total** and contains four inner columns. These inner columns indicate the number of nodes of all the BDDs necessary to calculate the probability and activity of the output ports. These inner columns are:
  1. The first column has been called **Regions**. It indicates the number of signals (except for input ports) whose probability and activity are necessary to compute the probability and activity of the outputs. It is equivalent to the number of partitions that have been done (that is why it has been called *regions*). If a circuit has the same number of output ports than regions, it means the circuit has not been partitioned at all
  2. The column called **BDD** shows the number of nodes of all probability BDDs of all the regions the circuit has been divided into
  3. The column **TFBDD** indicates the number of nodes of all TFBDDs of all the circuit partitions
  4. The column ***a*BDD** indicates the number of nodes of all *a*BDDs of all the circuit partitions
- The other group of columns indicates the size of the largest BDD (*Worst case*). Usually, this BDD corresponds to the largest partition. In consequence, since the information is related to just one region, there is no column called *regions* in this group. The three columns of this group indicate the number of nodes of each kind of BDD. Not always the largest probability BDD is in the same region of the largest activity BDD. In such case, the largest BDDs will be taken, independently if they belong to different regions.

E3.2.1 Circuit "max"

Table E3-2 shows the BDD sizes for circuit "max" having a bus width of 4 bits. For this circuit, both the gate level analysis and the exact-RTL analysis have the same number of regions (4). Since the circuit has four outputs, it means that the circuit could not be partitioned in those analyses.

bus width: 4 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Gate	4	74	1118	712	44	861	548
RTL	4	29	160	109	12	72	50
disjoint-RTL	7	38	193	121	6	32	19

Table E3-2: BDD sizes for circuit "max" depending on the analysis

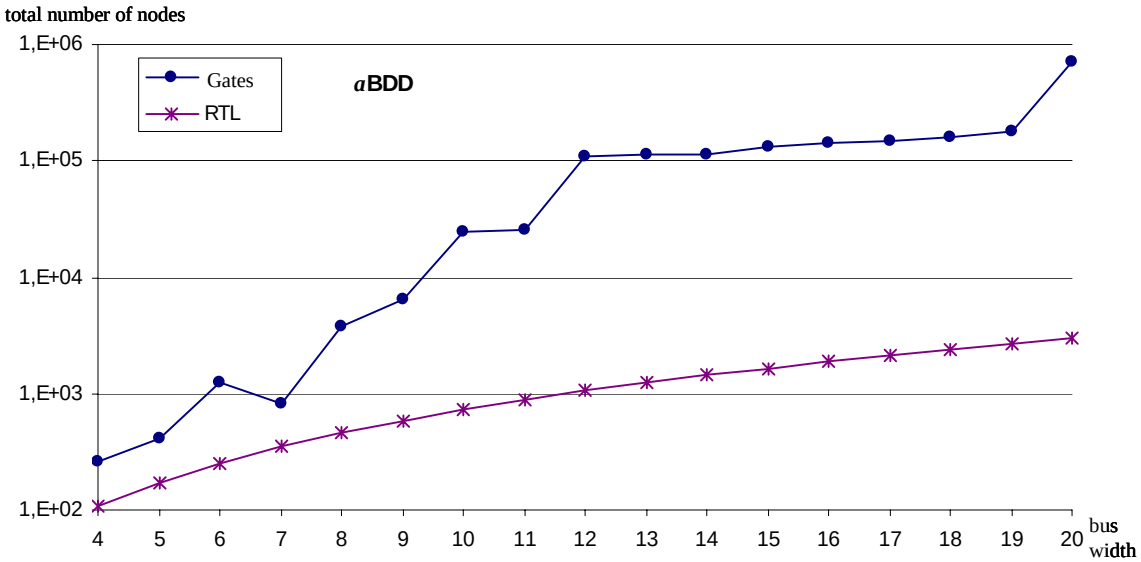
It can be observed from the table that, even though both gate and exact-RTL have the same regions, due to the better RTL ordering, the BDDs of the exact-RTL analysis are much smaller. For example, the worst case gate level activity BDDs are ten times larger than the RTL activity BDDs.

This circuit has a comparison operator (*bigger than*) between 2 four-bit operands. At RTL, the most significant bits have been taken first. That is: *A*(3), *B*(3), *A*(2), *B*(2), *A*(1), *B*(1), *A*(0), *B*(0). The identification of the operator and the vector indexes is an immediate task at RTL having the EHM (§E2.3.2); while it is hard to do at gate level. Therefore, this circuit is **easy to order at RTL**.

Consequently, considering that the results at exact-RTL are identical to those at gate level, due to the smaller size of the BDDs at RTL, the **RTL analysis is more convenient than the gate level**.

If the circuit is partitioned in disjoint regions, the largest BDDs are smaller, but not the total number of nodes of all BDDs. Therefore, it is not advantageous to partition the circuit due to the provoked error. It may seem strange that the total number of nodes is slightly larger for disjoint-RTL. The reason is that there are too many small regions, thus there are more intermediate BDD variables and also fewer simplification possibilities. The exact-RTL regions are manageable; thus, there is no reason to partition the circuit. As it was explained in section 3.4.3, **it is not advisable to perform exhaustive disjoint partition**; on the contrary, it is better to make the minimum disjoint partitions that provide controlled BDD sizes.

The evolution of the BDD sizes with the bus width has been analyzed. The original RTL circuit has been modified changing its operands bus width. *Synopsys Design Compiler* has been used to synthesize the RTL circuits into gate level circuits. Gate level and exact-RTL BDDs have been compared. Graphic E3-1 shows the evolution of the BDDs' total number of nodes with the bus width. Note that, due to the enormous differences, the graphic has been plotted in logarithmic scale.



Graphic E3-1: aBDD evolution with the bus width for circuit "max"

The graphic shows the variability of the gate level *a*BDDs. On the contrary, the exact-RTL BDDs are completely predictable. The numeric values of the graphic have been included in table AIII-1 of section AIII.1.1 of the annex, where a wider analysis is included (in Spanish).

The largest BDDs (worst case) of the exact-RTL analysis follow an arithmetic progression, whose n^{th}-terms are: $3 \cdot n$ for probability BDDs; $21 \cdot (n-1)+9$ for TFBDDs and $15 \cdot (n-1)+5$ for *a*BDDs.

Contrary to what happens at gate level, at RTL, BDD sizes follow a **lineal progression** that is manageable. For example, at RTL, the largest *a*BDD for 64-bit operands have 950 nodes, what is much smaller than the largest *a*BDD for 20-bit operands at gate level: 181869. Therefore, having this limited growth at RTL, it is not necessary to resort to disjoint partitioning.

In section AIII.1.1 of the annex (in Spanish) a wider analysis is included, in which it is shown that some of the gate level analysis could not be done due to excessive demand of memory resources. This fact supports the advantages of RTL estimation.

To conclude, this circuit has shown some of **advantages of the RTL analysis against the gate level analysis.** The RTL ordering produce so small BDDs that it is not necessary to resort to disjoint partitioning. Consequently, no error is produced. When the bus width is increased, the RTL BDDs follow a linear progression, while the gate level BDDs grow so much that in some cases it is not possible to carry out the analysis. Besides, the BDD size increment is not predictable, existing wide variations in size.

E3.2.2 Circuit "comparador"

Opposite to the previous circuit, this circuit has not much reconvergences. In fact, the RTL analysis performs the reconvergent partition, which is the same partition as the disjoint partition. Therefore, there is no error in the RTL analysis.

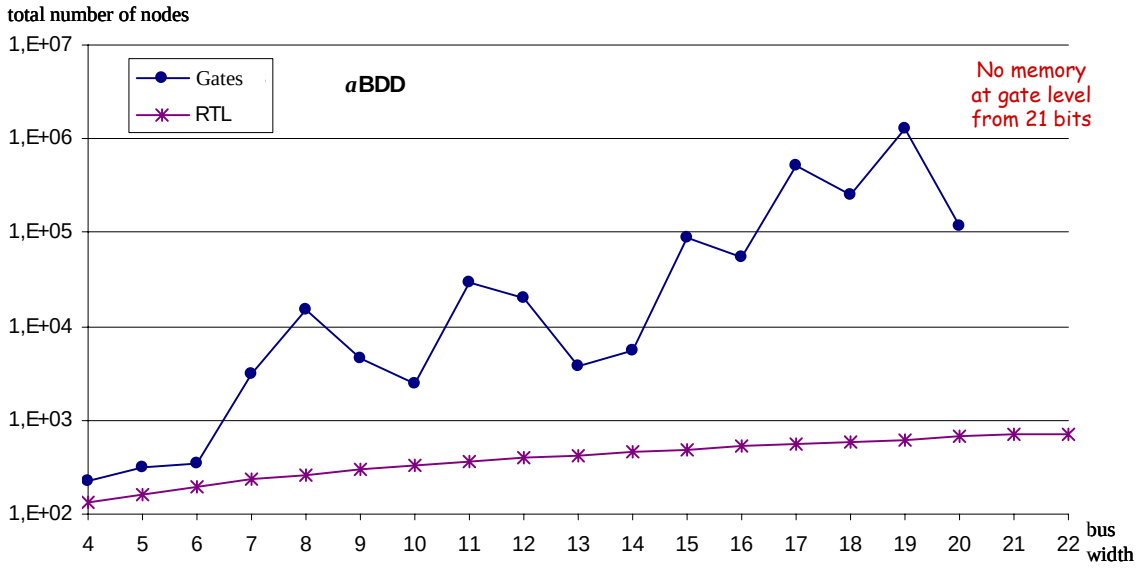
It would be sensible to think that at gate level the circuit would be partitioned in the same number of reconvergent regions. However, the gate level circuit cannot be partitioned in reconvergent regions. The reason is that **synthesis has incremented the reconvergences.**

Table E3-3 shows the BDD sizes depending on the analysis. As it has been already said, in this circuit, exact-RTL and disjoint-RTL analyses obtain the same regions. In annex section AIII.2 there is an analysis of the circuit partition (in Spanish).

bus width: 4 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Gate	3	51	329	230	19	135	96
RTL	12	48	204	132	4	18	12
disjoint-RTL	12	48	204	132	4	18	12

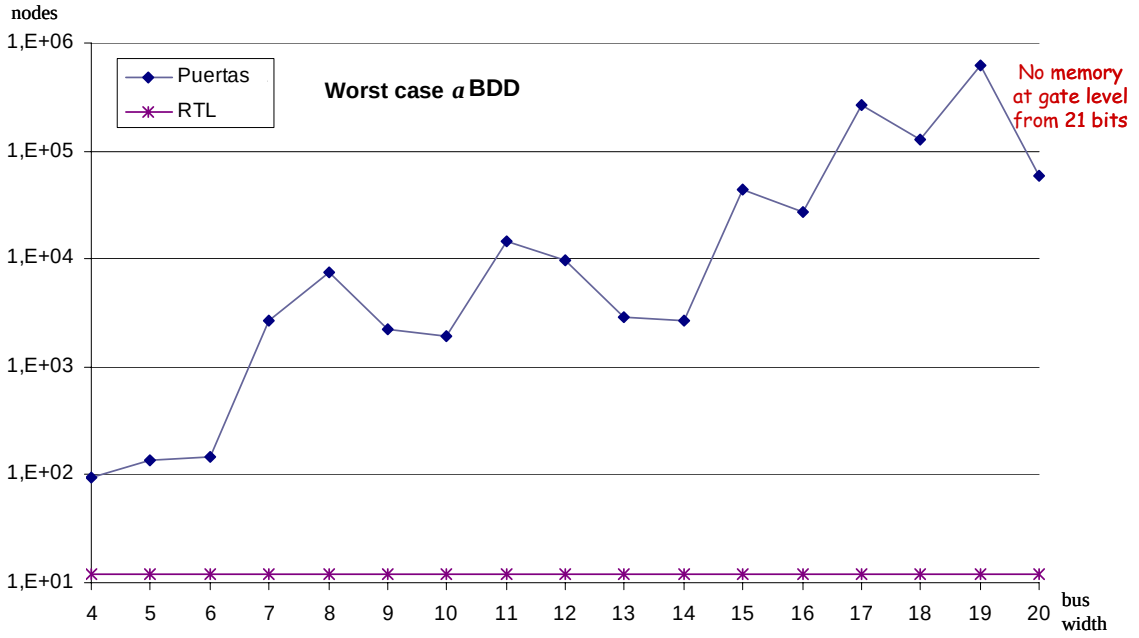
Table E3-3: BDD sizes for circuit "comparador" depending on the analysis

There are not great differences in size between the BDDs of table E3-3 because the circuit is too small to make visible the advantages of the RTL analysis. If the operand's bus width is increased, the differences are more notorious. Graphic E3-2 shows the evolution of the sum of the *a*BDD total of nodes with the bus width for gate and RTL analysis. Note that, again, the scale is logarithmic.



Graphic E3-2: aBDD evolution with the bus width for circuit "comparador"

The worst case *a*BDD evolution is shown in graphic E3-3.



Graphic E3-3: Worst case aBDD evolution with the bus width, circuit "comparador"

The numerical data of graphics E3-2 and E3-3 are in table AIII-4 in annex section AIII.2. From the tables it can be inferred that:

- The increment of the total number of nodes with the bus width is linear for the RTL analysis; while, for gate level, it is not only non-linear but also unpredictable.
- When the bus width is larger than 20 bits, the gate level circuit cannot be analyzed due to the large size of its activity BDDs.
- At RTL, the largest BDD (worst case) is constant even when the bus width is increased (BDD: 4, TFBDD: 18, *a*BDD: 12). On the contrary, at gate level, the size is variable and, in general, it rapidly augments with the bus width.

Therefore, due to the constant size of the worst case BDDs at RTL, any circuit "comparator" with any bus width can be addressed.

At gate level it is difficult to obtain an adequate BDD ordering because of the lack of high level information. In the gate level analyses, the variable ordering is taken depending on the signal arrangement after synthesis. Although this arrangement may have any structural meaning, it is rather arbitrary. Annex section AIII.2 (in Spanish) shows the difference in BDD size of the same gate level circuit using other gate arrangements (see table AIII-1).

The main conclusion of this experiment is the confirmation of the fact that **synthesis may increment circuit reconvergences**. As a result, the regions and the BDDs of gate level circuits would be larger than those at RTL. Therefore, it is another advantage of RTL analysis.

E3.2.3 Circuit "alu_peq"

This circuit is larger than the previous ones; therefore, when compared with the others having with the same bus width, the differences in BDD size are more notorious for this circuit.

Table E3-4 shows the number of nodes of the BDDs obtained from the circuit with four bit bus width depending on the analysis.

bus width: 4 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Gate	17	386	8999	5667	108	3958	2512
RTL	29	260	2171	1394	55	637	438
disjoint-RTL	31	251	1539	946	15	157	85

Table E3-4: BDD sizes for circuit "alu_peq" depending on the analysis

First, it can be observed from table E3-4 that there are fewer partitions at gate level. Thus, similarly to what happened in the previous circuit, **synthesis has increased the circuit reconvergences**. The number of activity BDD nodes is four times smaller for RTL than for gate level. If disjoint partition is carried out, the differences are wider, although it is not necessary due to the small size of this circuit.

The worst case BDD differences are larger. The activity BDDs are about 6 times smaller for exact-RTL than for gate level. Disjoint-RTL activity BDDs are about 25 and 30 times smaller than gate level activity BDDs.

E3.2.4 Circuit "alu_core"

The majority of the operations of this circuit are bit level; thus, they are not complex. Therefore, although this circuit has more input ports than the previous, the BDD sizes of this circuit are smaller. Table E3-5 shows the BDD sizes for the circuit "alu_core" having 8 bit bus width.

bus width: 8 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Gate	25	423	5010	3328	45	639	427
RTL	26	263	2472	1547	23	277	168
exhaustive disjoint-RTL	50	303	2448	1495	21	250	149
minimum disjoint-RTL	26	263	2392	1519	23	264	163

Table E3-5: BDD sizes for circuit "alu_core" depending on the analysis

Gate level BDDs are larger than RTL. The RTL analyses produce similar size BDDs. Due to the small difference; it is not advisable to perform the disjoint partition in this circuit. Since the majority of the operations are bit level, the BDD sizes do not increment significantly with the bus width. During the circuit analysis, it is possible to detect whether it may be useful to

perform a disjoint partition because EHM (§E2.3.2) provides the information relative to the region's number of dependences and the complexity of the operators.

E3.2.5 Circuit "addsub"

For an 8-bit bus width circuit, although it has fewer gates than the previous two circuits (see table E3-1) the BDDs are larger for this adder/subtracter. The main reason is that adders have an important number of XOR gates and these gates produce large BDDs. Table E3-6 shows the BDD sizes for an 8-bit bus width circuit.

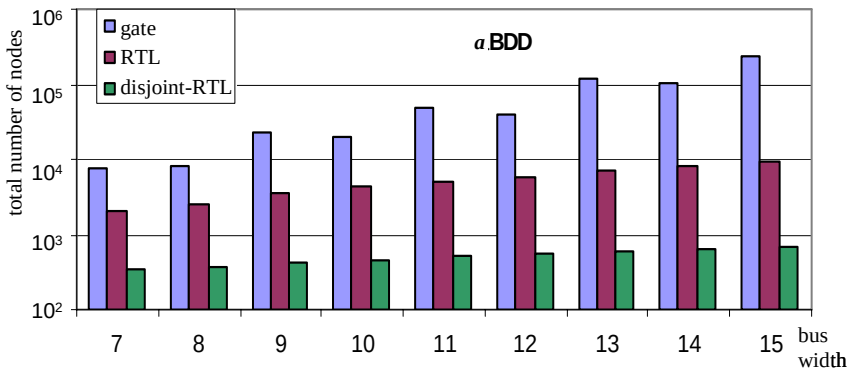
bus width: 8 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Gate	13	571	11177	8249	139	3837	2856
RTL	11	499	3599	2590	89	641	479
disjoint-RTL	45	187	645	375	5	19	13

Table E3-6: BDD sizes for circuit "addsub" depending on the analysis

As it can be deduced from table E3-6, the RTL circuit cannot be partitioned in reconvergent regions. The exact-RTL circuit has 11 regions; thus, since there are 11 output regions, the circuit has not been partitioned. The gate level circuit has been hardly partitioned because it has 13 regions and the additional two regions are small.

There are 45 disjoint regions. The BDD sizes are much smaller with this partition, especially the worst case BDDs. It is evident that it is not necessary to perform such an exhaustive partition with worst case *a*BDDs having no more than 13 nodes. Annex section AIII.5.1 shows details about the partitions.

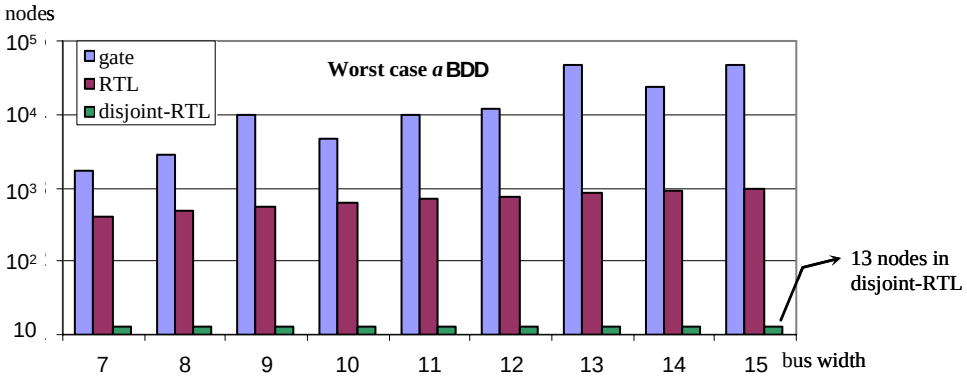
The operand bus width has been modified in order to study the variation of the BDD sizes. Graphic E3-4 shows the evolution of the total number of nodes of all *a*BDDs with the bus width. There are three bars in the graph: the first one is related to the gate level analysis, the second is the exact-RTL (with no disjoint partition) and the third is the disjoint-RTL analysis. Note that the graphic has been plotted in a logarithmic scale.



Graphic E3-4: aBDD evolution with the bus width, circuit "addsub"

As in the previous analyses, the size variations of the gate level analyses are not predictable. Besides, the number of nodes is an order of magnitude bigger for the gate level analysis.

Graphic E3-5 shows the evolution of the largest *a*BDD with the bus width. Note that, again, the scale is logarithmic.



Graphic E3-5: Worst case aBDD evolution with the bus width, circuit "addsub"

The numerical data of these graphs are in annex AIII.5.

The analysis of this circuit evidences the advantages of the better RTL ordering and, besides, the node reduction attained when disjoint region partition is carried out.

E3.2.6 Circuit "alu8051_simp"

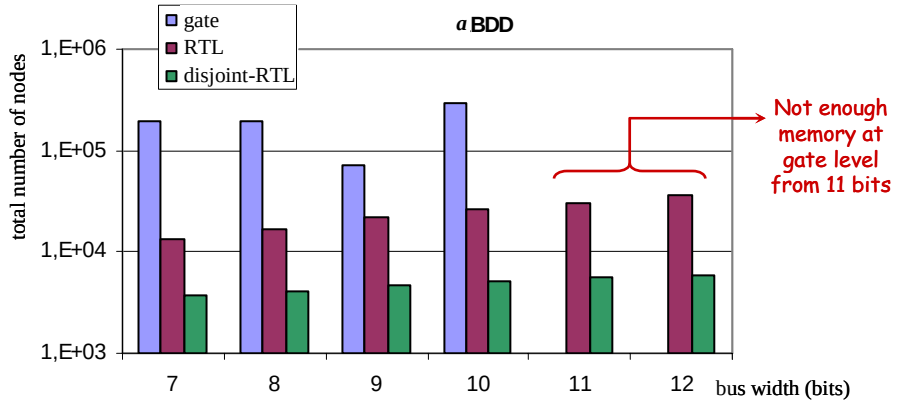
This circuit and the next are larger and have more input ports than the previous. For a bus width of 8 bits, table E3-7 shows the BDD number of nodes depending on the analysis.

bus width: 8 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Gate	21	2198	273365	198584	908	237571	175896
RTL	11	1423	26061	16813	224	4881	3215
disjoint-RTL	123	765	7068	4134	34	645	370

Table E3-7: BDD sizes for circuit "alu8051_simp" depending on the analysis

The RTL ordering shows notorious advantages over the gate level. The ordering consist in taking first the selection signal (§E2.6) and then the operands. The most significant indexes are taken before the less significant. At RTL is very easy to carry out this ordering but at gate level it is not.

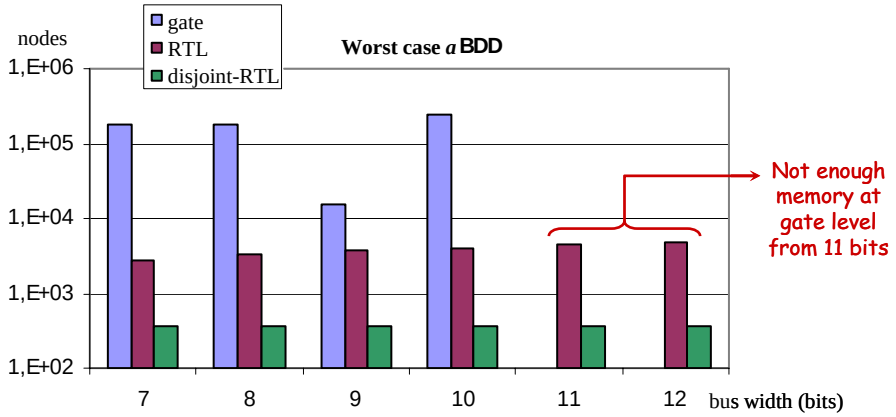
The operand bus width has been modified in order to study the variation of the BDD sizes. Graphic E3-6 shows the evolution of the total number of nodes of all *a*BDDs with the bus width. Note that the graphic has been plotted in a logarithmic scale.



Graphic E3-6: Total aBDD size evolution with the bus width, circuit "alu8051_simp"

As in the previous analyses, the size variations of the gate level analyses are not predictable. Besides, the number of nodes is an order of magnitude bigger for the gate level analysis.

Graphic E3-7 shows the evolution of the largest *a*BDDs with the bus width. Note that, again, the scale is logarithmic.



Graphic E3-7: Worst case aBDD evolution with the bus width, circuit "alu8051_simp"

Annex section AIII.6 includes the numeric data of these graphics.

These analyses evidence the advantages of RTL estimation. At gate level, due to the increment of the BDD sizes, from a certain number of bits, it is not possible to carry out the analysis.

Besides, the analyses show the benefits of disjoint partitions: they not only reduce the total BDD sizes but also maintain the largest BDDs (worst case) in constant size with the bus width.

E3.2.7 Circuit "alu8051"

This is the most complex circuit and the one with more input ports. This circuit cannot be partitioned in reconvergent regions. But, it can be partitioned in disjoint regions depending on how the circuit has been described or if a previous analysis is carried out (see annex AIII.7.1, in Spanish).

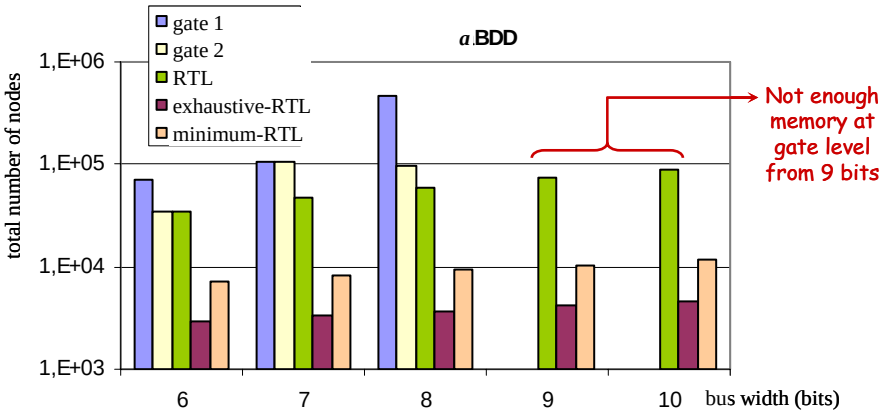
Table E3-8 shows the partitions and the BDD sizes for the circuit with an 8-bit bus width. Since at gate level and at RTL there are 11 output ports and 11 regions, it means that the circuit could not be partitioned. On the contrary, the circuit can be partitioned in disjoint regions. The table shows an exhaustive disjoint partition (§3.4.3) and a minimum disjoint partition. As it has been already said, it is not a good practice to perform exhaustive disjoint partitions; however it has been carried out for didactic reasons.

bus width: 8 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
Gate 1	11	4936	760118	467174	1176	275596	166061
Gate 2	11	2813	162315	95937	440	28070	16178
RTL	11	2819	99679	58909	425	18262	10807
exhaustive disjoint-RTL	596	1776	6638	3707	5	19	13
minimum disjoint-RTL	289	1470	17141	9290	57	1732	912

Table E3-8: BDD sizes for circuit "alu8051" depending on the analysis

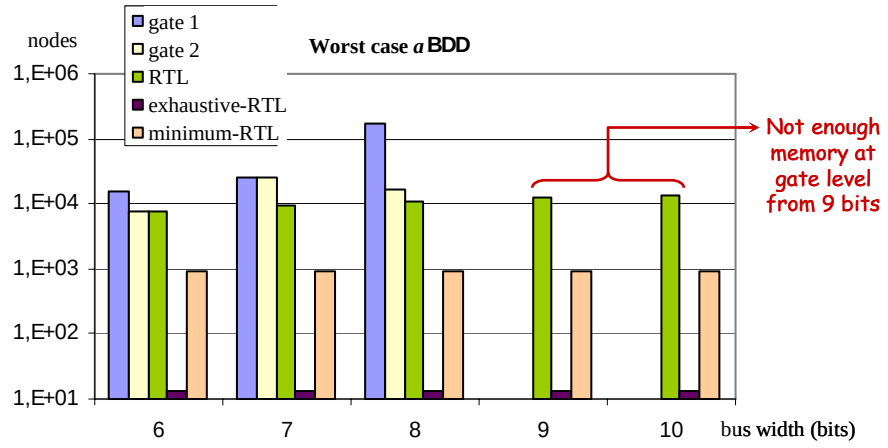
Two versions of the gate level circuit have been obtained with different synthesis options.

In order to analyse the evolution of BDD sizes, the number of bits of the operands have been changed. Graphic E3-8 shows the evolution of the total number of *a*BDDs nodes. Due to an excessive memory demand, from 9 bits, it was not possible to analyze any of the two gate level circuits.



Graphic E3-8: Total aBDD size evolution with the bus width for circuit "alu8051"

Graphic E3-9 shows the evolution of the largest *a*BDD with the bus width depending on the analysis.



Graphic E3-9: Worst case aBDD evolution with the bus width of circuit "alu8051"

Table AIII-17 of annex section AIII.7 shows (in Spanish) the numerical data of these graphics.

The analysis of this circuit reinforces the conclusions of the previous analysis:

- Due to the BDD size increment, usually, it is not possible to carry out the analysis at gate level when the circuit size and complexity rises.
- At gate level, when the operand bus width increments, the BDD growth is unpredictable
- The increment of the BDD size applying the exact-RTL analysis follows an arithmetical progression
- The size of the largest BDDs (worst case) applying the disjoint-RTL analysis does not vary with the bus width.
- Disjoint partitions not only significantly reduce the total number of nodes but also limit the largest BDDs; thus, allowing to analyze a circuit of any bus width. In large, complex circuits that cannot be partitioned in reconvergent regions, disjoint partition may be the solution to control the BDD growth.

E3.3 Error analysis

This section compares the activity results between the RTL models with and without disjoint partition. In section E2.4.3 was explained that, while disjoint partition produces exact probability values, activity values may have some error. The other models do not produce error, and should produce the same results than the simulative methods. But simulative methods do not produce exact results due to their dependence on the input vectors. Two simulative analyses over the same circuit may produce slightly different results. Generally, the differences are below 0.01 in absolute values. Therefore, in this Thesis, it has been considered that **differences lower than 0.01 do not constitute error**.

On the other hand, calculations applying TFBDDs and *a*BDDs give the same results. Hence, the only error to be analyzed is the error produced by the usage of disjoint regions.

In order to assure that the probabilistic method results correspond with the simulative (or dynamic) method, the probability and activity results of the first three circuits have been contrasted. The simulative results have been obtained through the circuit simulation assigning random input vectors that have been conditioned to certain probabilities and activities. As it has been said, the results depend on the specific vectors that are assigned; therefore, they are not exact unless the simulation lasts for an infinite time.

These circuits have been stimulated with vector depths of 16300 cycles. A large enough size for the circuit dimensions. Due to the small size of the circuits, compaction methods were not needed [81], [77]. The differences have been lower than 0.01.

Next, the method to analyze the error is explained. Afterwards, the error due to disjoint partitions is analyzed for each circuit.

E3.3.1 Disjoint partition error analysis

Analyzing the error is not an easy task because it depends on the probability and activity values of the inputs. Consequently, there are an infinite number of combinations, which will give a different error.

Generally, other works studied the error assigning certain values to the inputs' probabilities and activities. For example, Schneider [119] assigned the values P_{in}=0.5 y a_{in}=0.25 to all the inputs. Chou [28] assigned P_{in}=0.5 and two different activity values: a_{in1}=0.1 ; a_{in2}=0.26. In other works it was not specified, but because the results are compared with Schneider's, it could be assumed that they used the same values.

This Thesis has carried out a wider analysis. The analysis has not been performed with a few of specific probability and activity values; instead the values of a probability-activity mesh have been assigned to the inputs. Since not all the probability-activity pairs are possible (equation E1.6), the assigned pairs have to be inside the triangle shown in figure E1-3 (which has been copied in figure E3-10). There are 31 different probability values, with steps of 0.03125; and 64 activity values (steps: 0.015625). There are 1024 pairs inside the triangle; thus, there will be 1024 different activity estimations for a circuit.

Inside the triangle there is a curve that indicates the pairs which do not have temporal correlations and, thus, no error is produced. The pairs near this curve will produce low errors (condition 3 of section E2.4.3).

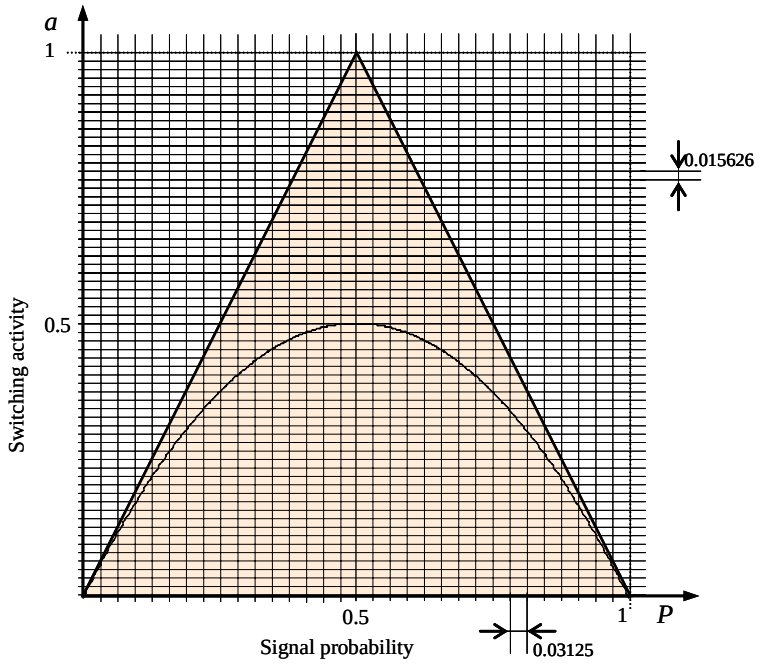


Figure E3-10: Mesh of probability-activity pairs assigned to the circuit inputs

For the error calculation, it will be considered the absolute error, that is:

$$e = a^{disj} - a \tag{3.1}$$

Where a^{disj} is the switching activity resulting from disjoint partition and a is the activity resulting from the analysis without disjoint partitions. Activity a can be calculated either by the gate level analysis or exact-RTL analysis because they both give the same result.

Taking the absolute error and not the relative error is a common practice ([119], [80], [7]). Due to the fact that the activity is a probability, the relative errors for very small values are large, but they do not always have a practical meaning[74]. Even the statistics of test vector being generated randomly with low activities (0,016) may produce relative errors of 25%.

As it has been already said, in this Thesis, absolute errors below 0.01 will be considered negligible. Absolute errors near 0.01 have been observed with the simulative analysis[75].

Unfortunately, since this Thesis does not work at the same level as the previous works and thus, the circuits analyzed are different, **the error values of this Thesis are not comparable with earlier proposals**. Besides, the circuit internal nodes are different at RTL (usually, there are less nodes at RTL). Generally, the other works compare the activities of all circuit internal nodes, because at gate level they are perfectly defined. Therefore, the error statistics (mean, variance ...) are calculated over all the internal nodes. As a result, the mean error is lower because the error of the nodes nearer to the inputs is inferior. In this Thesis, the error statistics are taken only over the circuit outputs, but statistics are extracted from the 1024 estimations arising from the assignment of the mesh of pairs (figure E3-10). This way of analyzing the circuit and taking the statistics has two consequences that worsen the error statistics:

- The error values at circuit outputs are expected to be higher due to their depth. They may have a more complex logic and the error may increment because of the propagation of approximate activities through the circuit.

[74] For example, if $P(A='1')_{estim}$=0.9 and $P(A='1')_{exact}$=0.95; the relative error will be $e_{rel}(A='1')$=(0.95–0.90)/0.95 = 0.053. If from the same example, the probability of being zero would be calculated: $P(A='0')_{estim}$=0,1 y $P(A='0')_{exacta}$=0.05. Therefore, $e_{rel}(A='0')$=(0.05–0.10)/0.05 = 1. Note the differences, even when it is the same error.

[75] Errors around this magnitude have been observed having more than 10^4 vectors

- The assignment of all the mesh of pairs (figure E3-10) analyzes the circuit in a wide number of different conditions. Some of these conditions are extreme and improbable to happen in normal circuit operation; while in the other proposals, only a few probability-activity pairs are assigned.

In order to have an evaluation of the error, table E3-9 includes the error statistics of different proposals. The experiments have been made over the ISCAS'85 circuits [15]. The second column (*Gates*) shows the number of logic gates of the circuit. The third column (*In ports*) indicates the number of input ports of the circuit. These numbers can be compared with those of table E3-1, which shows the circuits used for the experiments of this Thesis. Except for the last two circuits, the circuits have similar size. Nevertheless, these two last circuits (*c3540* and *c6288*), as well as other ISCAS'85 circuits, have been described without using XOR gates. This fact provokes the number of gates to increment because they are arithmetic circuits and these circuits use many XOR gates. Three or four gates (AND, OR and inverters) are needed to implement an XOR gate. Circuit *c3540* is an ALU and circuit *c6288* is a multiplier.

Note that some of the errors of table E3-9 are above 0.01. These data could be contrasted with those of table E3-28 of section E3.4.5, where a summary of the errors of the proposed method is shown. In addition, table AIII-18 of the annex section AIII.8 (in Spanish) is an extension of table E3-28.

Circuit	Gates	In ports	*Schneider* [119]	*Marculescu* [80]			*Bhanja* [8]		
			μ	μ	σ	e_{max}	μ	σ	e_{max}
c432	160	36	0.016	0.028	0.04	0.21	0.002	0.03	0.197
c499	202	41	-	0.013	0.01	0.062	0	0	0.006
c880	383	60	0.006	0.013	0.02	0.069	0.001	0.01	0.066
c1355	546	41	0.005	0.004	0	0.003	0.001	0.02	0.124
c1908	877	33	0.01	0.009	0.02	0.131	0.001	0.01	0.099
c3540	1669	50	0.014	0.03	0.04	0.201	0.005	0.04	0.252
c6288	2416	32	0.023	0.014	0.02	0.089	0.006	0.02	0.318

Table E3-9: Error statistics of different proposals

Before going through the error analysis, the conditions that minimize the error will be reminded (§E2.4.3):

1. The selection signal should be independent of the alternatives
2. The selection signal should have low activity
3. Reconvergent signals should have low temporal correlation
4. The alternatives should have a low rate of common sources compared to the independent sources

The first condition is considered necessary to partition the circuit in disjoint regions. This condition and the fourth are **structural conditions**. That is, the circuit structure determines whether the conditions are fulfilled or not. The second and third conditions are **extrinsic conditions**, because they depend on the probability and activity values of the dependences; thus, the extrinsic condition fulfillment depends on the circuit input values. That is the reason why a mesh of probability-activity pairs has been used to test the circuit error. These pairs allow to test the circuit in totally different input conditions. This fact can be observed in the experimental results because the error will vary depending on the pair taken. During the analysis, it is possible to know whether the conditions meet; therefore, the reliability of the partition can be assessed and even evaluate if it is advisable to partition the circuit.

Besides, it is important to remark that **exhaustive disjoint partitions should be avoided**. The number of disjoint partitions should be as low as possible.

In the following subsections, the error of each circuit will be analyzed.

E3.3.1.1 Circuit "max"

Section E3.2.1 showed that the RTL ordering for this circuit provided an adequate order. With this ordering, the activity BDDs increment linearly with the bus width. Besides, this increment is low and the disjoint partition does not provide any significant advantage compared with the exact-RTL analysis.

Disjoint partitions should produce moderately large BDDs, avoiding small BDDs. A minimum of a thousand of nodes may be considered addecuate, like the activity BDDs of circuit "alu8051" (§4.2.7). Nevertheless, the operands of "max" circuit should be 70 bits width to reach a thousand of *a*BDD nodes.

Consequently, it is not advisable to partition the circuit in disjoint regions. The decision as to whether a circuit should be partitioned can be taken during the analysis because EHM provides the information relative to the circuit structure and the operators involved. In this case, seeing that it is just an operator "bigger than", the disjoint partition is not necessary. Nevertheless, annex section AIII.1.2 (in Spanish) studies the error for the case of a circuit's exhaustive disjoint partition. This study is merely academic, performed in order to prove the proposed conditions for the disjoint partition (§3.4.3).

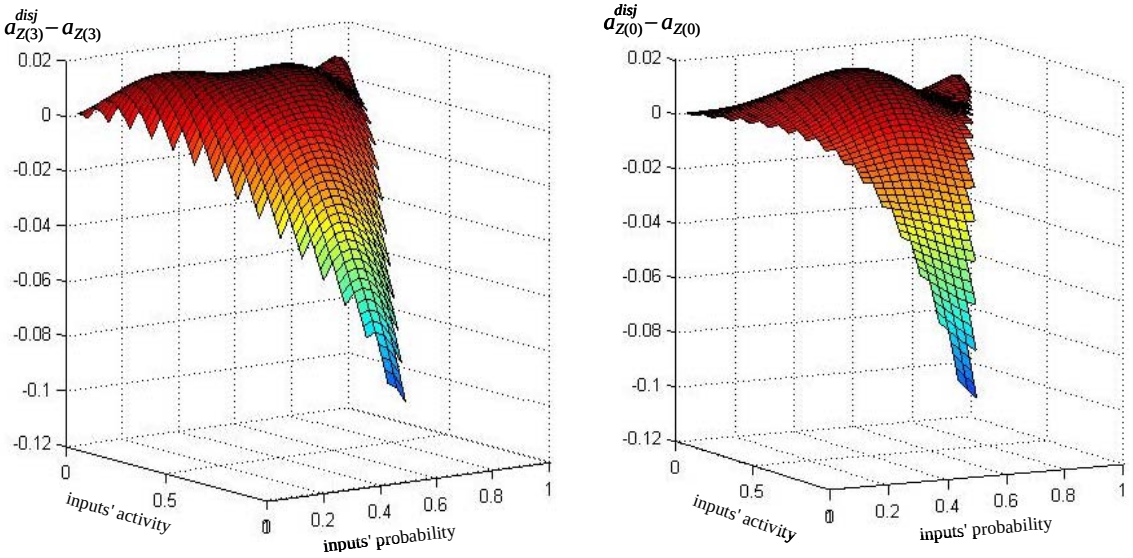
E3.3.1.2 Circuit "comparador"

In this circuit, the disjoint regions and the reconvergent regions are identical. Therefore, **no error is produced in the activity calculation**.

E3.3.1.3 Circuit "alu_peq"

Despite the small size of this circuit, the disjoint partition offers an interesting node reduction (§E3.2.3 y annex AIII.3). When the circuit is partitioned in disjoint regions, the result signal *Z*[3:0] has an error. Signal *Flags*(2) has a negligible error, while signal *Flags*(1) and *Flags*(0) has no error.

The error distribution of the four bits of *Z* is similar. Graphic E3-10 shows the error distribution for signals *Z*(3) and *Z*(0) for all the mesh of probability-activity pairs (remind figure E3-10).



Graphic E3-10: Error distribution of signals Z(3) and Z(0) of circuit "alu_peq" for all the range of possible probabilities and activities of the inputs (same value for all the inputs)

Table E3-10 shows the mean errors, the standard deviations and the maximum positive and negative errors for the output ports. In order to calculate the mean errors, the error values have been taken without sign (absolute value).

Signal	μ	σ	e_{max-}	e_{max+}
Z(3)	0.015	0.016	-0.109	+0.016
Z(2)	0.015	0.017	-0.109	+0.016
Z(1)	0.014	0.017	-0.109	+0.019
Z(0)	0.012	0.016	-0.109	+0.016
Flags(2)	0.0001	0.0001	-0.0004	+0.0004
Flags(1)	0	0	0	0
Flags(0)	0	0	0	0

Table E3-10: Mean error, standard deviation and maximum errors for each output port. Input conditions are the same for all inputs

The distribution of the error absolute values is shown in table E3-11.

distribution	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$0.02 < \|e\| < 0.03$	$0.03 < \|e\| < 0.04$	$0.04 < \|e\| < 0.05$	$\|e\| > 0.05$
Z(3)	48.4 %	33.9 %	5.1 %	3.7 %	2.9 %	6.0 %
Z(2)	50.0 %	32.6 %	4.8 %	3.8 %	2.8 %	6.0 %
Z(1)	51.2 %	31.9 %	5.1 %	3.3 %	2.6 %	5.9 %
Z(0)	64.4 %	21.5 %	3.9 %	3.2 %	2.1 %	4.9 %
Flags(2)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(1)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(0)	100 %	0 %	0 %	0 %	0 %	0 %

Table E3-11: Error distribution. Input conditions are the same for all inputs

Signal *Flags* does not have error, while signal *Z* has. As it has been said, errors below 0.01 are considered negligible. It can be observed from table E3-11 that, for signal *Z*, the group of errors smaller than 0.01 is the most numerous, having about the half of the total number of cases. More than the 80% of the error is under 0.02.

A more detailed analysis shows that the maximum positive error is not big, because it is below 0.02 (table E3-10). The mean error is not higher than 0.015. The (negative) errors are higher when the activities are higher. The maximum error value is when $P_{in} = 0.5$ and $a_{in} = 1$. These input conditions imply that at every time step every input port bit changes its value. This situation is **unreal** for a circuit in normal operation. This situation provokes that the extrinsic conditions to perform the disjoint partitions do not meet (§E2.4.3). These conditions are:

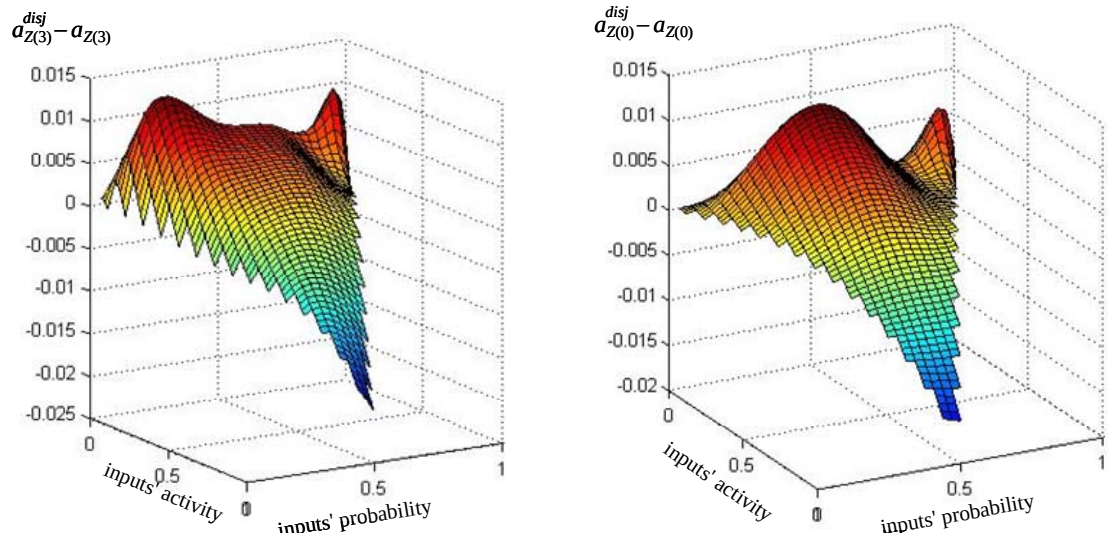
- Condition 2: The selection signal has low activity
- Condition 3: Reconvergent signals have low temporal correlation

Annex section AIII.3 (in Spanish) perform a detailed analysis about the error causes and the disjoint partition. Now, the influence of these conditions will be proved by means of two experiments.

A first experiment **limits the selection signal bits activity** to 0.1. The other signal probability-activity pairs cover the whole range of the mesh (figure E3-10). Therefore, in one of the estimations, the bits of signal *Z* will still have those unreal input conditions: $P_{in} = 0.5$ and $a_{in} = 1$. Nevertheless, selection signal (*Code*) will have real operation values.

Limiting the activity of the selection signal bits to 0.1 implies that the selection signal, considered as a whole (not bit by bit), has an activity $a_{code} = 0.27$. Thus, on average it will change its value every 3 or 4 cycles[76], what may be a reasonable value.

Graphic E3-11 shows the error distribution of signals *Z*(3) and *Z*(0) when the activity of signal *Code* is limited. Compare these graphics with those of graphic E3-10, in which the activity of signal *Code* was not limited.



Graphic E3-11: Error distribution of signals Z(3) and Z(0) of circuit "alu_peq" for all the range of possible probabilities and activities of the inputs, but the activity of selection signal bits are limited to 0.1

Table E3-12 shows the mean errors, the standard deviations and the maximum positive and negative errors for the output ports. In order to calculate the mean errors, the error values have been taken without sign (absolute value).

Signal	μ	σ	e_{max-}	e_{max+}
Z(3)	0.0051	0.0037	-0.021	+0.012
Z(2)	0.0050	0.0037	-0.021	+0.013
Z(1)	0.0049	0.0039	-0.020	+0.015
Z(0)	0.0037	0.0033	-0.019	+0.010
Flags(2)	0.0001	0.0001	-0.0004	+0.0004
Flags(1)	0	0	0	0
Flags(0)	0	0	0	0

Table E3-12: Mean error, standard deviation and maximum errors for each output port. The activity of selection signal bits are limited to 0.1

The distribution of the error absolute values is shown in table E3-13.

[76] The activity of signal *Code* when considered as a whole is calculated by the equation: $a_{code} = a_{code(0)} + a_{code(1)} + a_{code(2)} - a_{code(0)} \cdot a_{code(1)} - a_{code(0)} \cdot a_{code(2)} - a_{code(1)} \cdot a_{code(2)} + a_{code(0)} \cdot a_{code(1)} \cdot a_{code(2)}$

distribution	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$0.02 < \|e\| < 0.03$	$0.03 < \|e\| < 0.04$	$0.04 < \|e\| < 0.05$	$\|e\| > 0.05$
Z(3)	90.3 %	9.5 %	0.2 %	0 %	0 %	0 %
Z(2)	89.9 %	10 %	0.1 %	0 %	0 %	0 %
Z(1)	87.5 %	12.4 %	0.1 %	0 %	0 %	0 %
Z(0)	95.6 %	4.4 %	0 %	0 %	0 %	0 %
Flags(2)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(1)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(0)	100 %	0 %	0 %	0 %	0 %	0 %

Table E3-13: Error distribution. The activity of selection signal bits are limited to 0.1

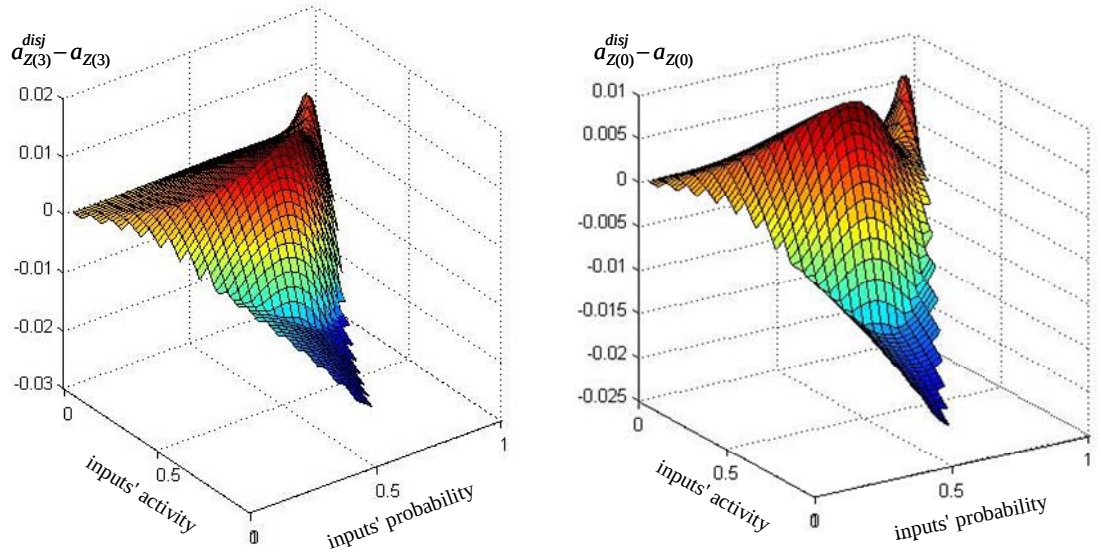
The comparison between these graphics and tables with the previous confirms the influence of the activity of the selection signal on the error. Since the activity of the selection signal is a known data when the partition is being performed[77], **the influence of the activity could be assessed in order to decide whether the partition should be carried out**.

The maximum error also occurs in the improbable case when $P_{in} = 0.5$ y $a_{A(i)} = a_{B(i)} = 1$; but know, it is much smaller.

The second experiment **limits the selection the temporal correlation of the reconvergent signals**. Then, the input ports *A* and *B* will have their activity limited to:

- When $a - a^{ti} > 0.1$ the activity will be $a = a^{ti} + 0.1$
- When $a^{ti} - a > 0.1$ the activity will be $a = a^{ti} + 0.1$

The selection signal (*Code*) will be assigned the activity related to the corresponding mesh of pair (without any limitation), including the improbable case when $P_{in} = 0.5$ y $a_{in} = 1$. The error distribution of signals *Z*(3) and *Z*(0) is shown in graphic E3-12.



Graphic E3-12: Error distribution of signals Z(3) and Z(0) of circuit "alu_peq" for all the range of possible probabilities and activities of the inputs, but with limited temporal correlation for inputs A and B

Table E3-14 shows the mean errors, the standard deviations and the maximum positive and negative errors for the output ports. In order to calculate the mean errors, the error values have been taken without sign (absolute value).

[77] The circuit analysis is performed from inputs to outputs; therefore, the probability and activity of the dependences is known

Signal	μ	σ	e_{max-}	e_{max+}
Z(3)	0.0065	0.0058	-0.022	+0.011
Z(2)	0.0065	0.0058	-0.022	+0.011
Z(1)	0.0064	0.0058	-0.022	+0.011
Z(0)	0.0057	0.0053	-0.022	+0.010
Flags(2)	0	0	0	0
Flags(1)	0	0	0	0
Flags(0)	0	0	0	0

Table E3-14: Mean error, standard deviation and maximum errors for each output port. The temporal correlation of inputs A and B has been limited

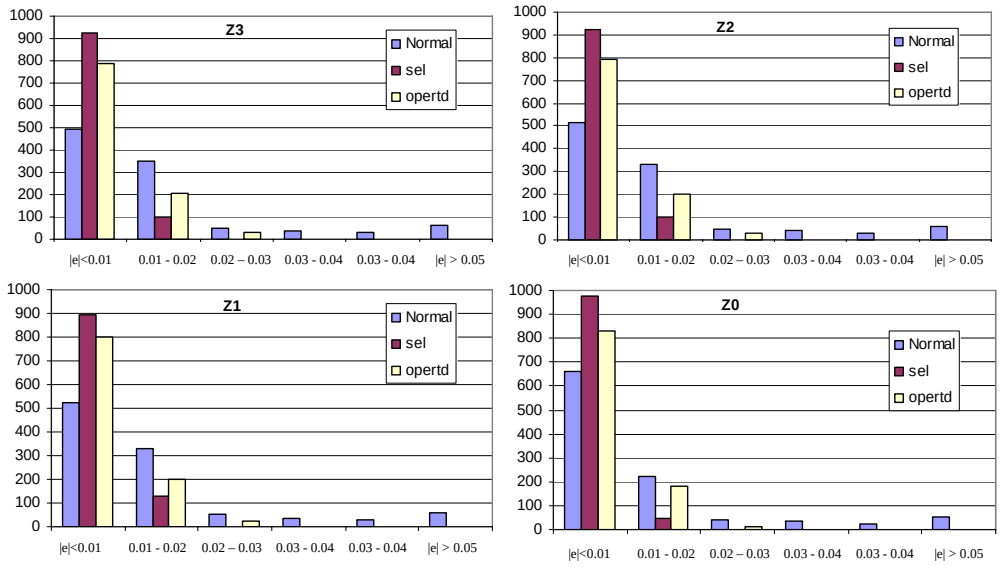
The distribution of the error absolute values is shown in table E3-15.

distribution	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$0.02 < \|e\| < 0.03$	$0.03 < \|e\| < 0.04$	$0.04 < \|e\| < 0.05$	$\|e\| > 0.05$
Z(3)	77.2 %	19.9 %	2.9 %	0 %	0 %	0 %
Z(2)	77.7 %	19.6 %	2.7 %	0 %	0 %	0 %
Z(1)	78.1 %	19.5 %	2.4 %	0 %	0 %	0 %
Z(0)	81.0 %	18.0 %	1.0 %	0 %	0 %	0 %
Flags(2)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(1)	100 %	0 %	0 %	0 %	0 %	0 %
Flags(0)	100 %	0 %	0 %	0 %	0 %	0 %

Table E3-15: Error distribution. The temporal correlation of inputs A and B has been limited

These results are also better than the initial ones. Nevertheless, the activity of the selection signal has not been limited; thus, the highest error also happen in the improbable situation when $P_{in} = 0.5$ y $a_{Code(i)} = 1$.

Graphic E3-13 compares the error distribution of signal *Z* bits for the three experiments that have been carried out. The first bar corresponds to the initial experiment (*normal*), where input signals take all possible mesh pair values. The second bar (*sel*) corresponds to the experiment in which the activity of selection signal (*Code*) was limited. The third bar (*opertd*) refers to the experiment where the temporal correlations of the reconvergent signals (operands *A* and *B*) were limited.



Graphic E3-13: Comparison of the error distribution for the 3 experiments: 1) normal, i.e. no limitation 2) selection signal activity limited (sel) 3) reconvergent signal temporal correlation limited

The last two experiments, which have limited either the selection signal activity or the reconvergent signals' temporal correlation, have been applied separately. In both experiments, the maximum errors are still around the improbable input conditions: P_{in} = 0,5 and a_{in} = 1 for the signals that have not been limited. Consequently, if these limitations were applied simultaneously, the error would be lower.

From the analysis, it can be concluded that:

- If a non-exhaustive disjoint partition is applied (see annex AIII.3, in Spanish) in most of the cases the error is acceptable (lower than 0.02). Besides, the disjoint partition produces small size BDDs (§E3.2.3).
- There are few cases with considerable errors. However, the input conditions of these cases are improbable and can be detected during the estimation process. Therefore, the error magnitude can be assessed and then, decide whether the disjoint partition is advisable.
- The inputs conditions that may produce significant errors provoke that some disjoint partition requirements (2 and 3) do not meet (§E2.4.3), which are:
    - A not too high selection signal activity
    - A not too high temporal correlation of the reconvergent signals

The results of this section suggest that for this circuit, the minimum disjoint partition provides with low error results and a small size BDDs. Nevertheless, due to the circuit size, it is not necessary to partition the circuit in disjoint regions.

E3.3.1.4 Circuit "alu_core"

Section E3.2.4 showed that the disjoint partition of this circuit produces slightly smaller BDDs than the exact-RTL analysis. Therefore, since the partition does not imply a significant benefit, it is not necessary to perform the partition. Besides, as the majority of the operators are bit level, the disjoint region partition does not provide a substantial advantage when the operand bus width is incremented.

Nevertheless, the minimum disjoint partition does not produce error. For all the cases, the error is lower than 0.0001, which is a value much lower than the maximum negligible error considered in this Thesis (0.01).

Solely for academic reasons, in annex section AIII.4 (in Spanish), the exhaustive disjoint partition error is analyzed. The annex shows that this error is not significant, only in some extreme conditions the error may be considered.

E3.3.1.5 Circuit "addsub"

Section E3.2.5 showed that the disjoint region partition produce smaller BDDs. This section will analyze if the error caused by the disjoint regions is acceptable.

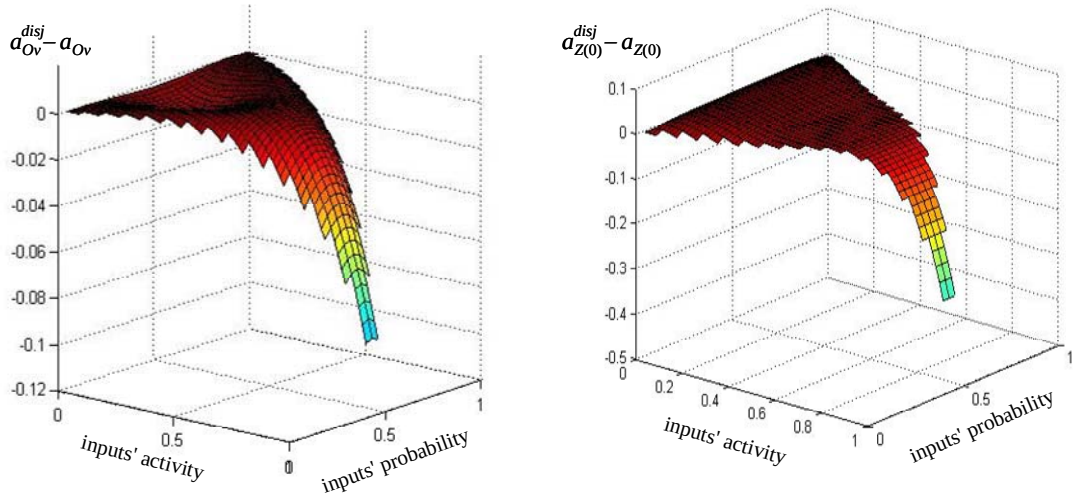
The same kind of analysis has been carried out, in which the inputs take all the mesh probability-activity pairs (figure 4-10). The mean error, the standard deviation and the maximum positive and negative errors are shown in table E3-16.

Signal	μ	σ	e_{max-}	e_{max+}
Z(7)	0.0003	0.0005	-0.0002	0.002
Z(6)	0.0004	0.0009	-0.005	0.003
Z(5)	0.0005	0.0010	-0.005	0.003
Z(4)	0.0007	0.0010	-0.002	0.004
Z(3)	0.0010	0.0014	-0.003	0.006
Z(2)	0.0017	0.0019	-0.004	0.008
Z(1)	0.0042	0.0043	-0.020	0.011
Z(0)	0.0206	0.0442	-0.50	0.026
Co(1)	0.00005	0.0001	-0.003	0.0001
Co(0)	0.0004	0.0006	-0.004	0.0006
Ov	0.0072	0.0146	-0.125	0.0009

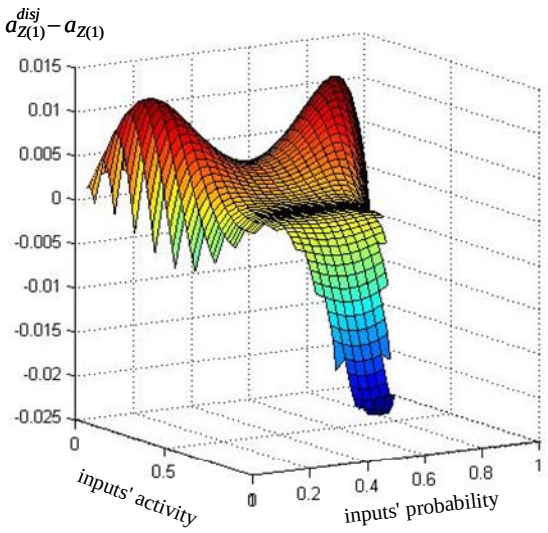
Table E3-16: Mean error, standard deviation and maximum errors for each output port. Input conditions are the same for all inputs

The only signals having non-negligible errors are *Z*(1), *Z*(0) and *Ov*. The other signals have errors between -0.01 and 0.01.

For signal *Z*(0) the maximum negative error is high: 0.5. However, this error also happens in improbable conditions, when P_{in} = 0.5 ; a_{in} = 1. Graphics E3-14 and E3-15 show the error distribution of these signals (*Ov*, *Z*(0), *Z*(1)). It can be observed that it is in the areas of maximum activity where the highest errors are produced.



Graphic E3-14: Error distribution of signals Ov and Z(0) of circuit "addsub" for all the range of possible probabilities and activities of the inputs (same value for all the inputs)



Graphic E3-15: Error distribution of signal Z(1) of circuit "addsub" for all the range of possible probabilities and activities of the inputs (same value for all the inputs)

The error distribution is shown in table E3-17. There only are significant errors in the distributions of signals *Z*(0) and *Ov*.

distribution	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$0.02 < \|e\| < 0.03$	$0.03 < \|e\| < 0.04$	$0.04 < \|e\| < 0.05$	$\|e\| > 0.05$
Z(7)	100 %	0 %	0 %	0 %	0 %	0 %
Z(6)	100 %	0 %	0 %	0 %	0 %	0 %
Z(5)	100 %	0 %	0 %	0 %	0 %	0 %
Z(4)	100 %	0 %	0 %	0 %	0 %	0 %
Z(3)	100 %	0 %	0 %	0 %	0 %	0 %
Z(2)	100 %	0 %	0 %	0 %	0 %	0 %
Z(1)	90.3 %	8.8 %	0.9 %	0 %	0 %	0 %
Z(0)	48.1 %	22.7 %	20.3 %	1.8 %	0.9 %	6.2 %
Co(1)	100 %	0 %	0 %	0 %	0 %	0 %
Co(0)	100 %	0 %	0 %	0 %	0 %	0 %
Ov	85.4 %	5.5 %	2.6 %	2.2 %	1.4 %	2.9 %

Table E3-17: Error distribution of circuit "addsub"

Nevertheless, the **error of signal** *Z*(0) **is easy to avoid**. Signal *Z*(0) only depends on four input ports: *A*(0), *B*(0), *Addsub*, *Ci*; therefore, since the BDDs will not be large, no disjoint partition should be made for signal *Z*(0). The decision about not partitioning signal *Z*(0) can be made during the analysis because the EHM provides the information about the number of dependences (§E2.3.2). The decision about not partitioning signal *Z*(0) is in agreement with avoiding exhaustive disjoint partitions (§E2.4.3). With this decision, the error of signal *Z*(0) is completely eliminated and the resulting BDDs will have similar size. Table E3-18 compares the resulting BDDs when signal *Z*(0) is partitioned and when it is not.

Bus width: 8 bits	Total				Worst case		
	Regions	BDD	TFBDD	*a*BDD	BDD	TFBDD	*a*BDD
disjoint-RTL	45	187	645	375	5	19	13
disjoint-RTL-without *Z*(0)	43	179	621	363	5	19	13

Table E3-18: BDD size comparison when Z(0) is partitioned and when it is not

The table shows that, when signal *Z*(0) is not partitioned, the total number of BDD nodes is even lower.

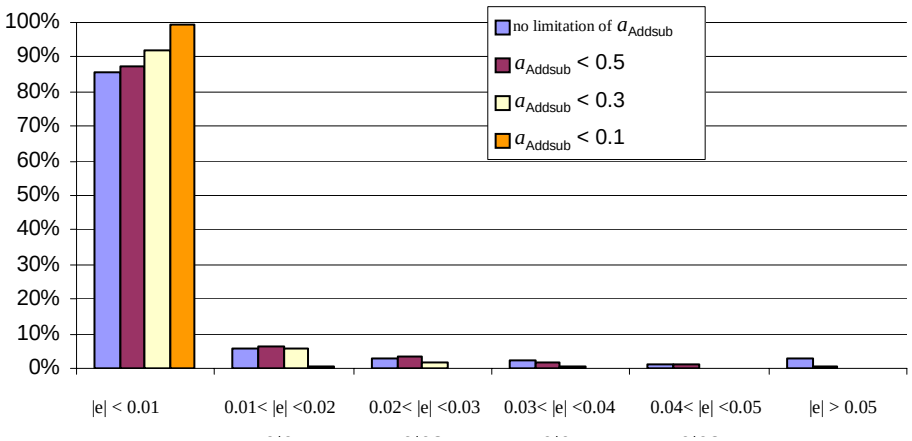
In a similar way, the **error of signal** *Z*(1) could be **avoided** if it was not partitioned. Since signal *Z*(1) only has a few of dependences, the resulting BDDs will be small.

Thus, now, the only signal having error is *Ov*. It can be observed from graphic E3-14 that its error is restricted to the areas of high activity. For the other areas the error is small: in the 85.4% of the cases the error is lower than 0.01. And in the 90.9% the error is lower than 0.02. It is a similar case as in circuit "alu_peq" (circuit 3), where the cases of high error are unreal situations (annex section AIII.3.1, in Spanish). When signal *Addsub* has high activity the error increments. If a similar experiment to the one of circuit "alu_peq" is carried out the error will decrease. Table E3-19 shows the error statistics of some experiments where the activity of *Addsub* has been limited. The input conditions that produce the maximum errors have been also included. In all cases, the highest error is produced when $P_{in} = 0.5$; $a_{in} = 1$

Error of signal *Ov*	μ	σ	Negative maximum error (e_{max-})			Positive maximum error (e_{max+})		
			P_{in}	a_{in}	e_{max-}	P_{in}	a_{in}	e_{max+}
no limitation of a_{Addsub}	0.0072	0.0146	0.5	1	-0.125	0.781	0.266	0.0009
a_{Addsub}< 0.5	0.0053	0.0084	0.5	1	-0.062	0.781	0.266	0.0009
a_{Addsub}< 0.3	0.0038	0.0052	0.5	1	-0.037	0.781	0.266	0.0009
a_{Addsub}< 0.1	0.0018	0.0021	0.5	1	-0.012	0.187	0.219	0.0004

Table E3-19: Mean error, standard deviation and maximum errors for each output port. The input conditions that produce the maximum errors are included. Input conditions are the same for all inputs except for a_{Addsub}, which has been limited

The error distribution of signal *Ov* depending on the limitation of *Addsub* is shown in graphic E3-16.



Graphic E3-16: Comparison of the error distribution of signal Ov depending on the limitation of the activity of Addsub

Graphic E3-16 shows that the majority of the errors are lower than 0.01. And when $a_{Addsub} \leq 0.1$ it may be considered that there is no error. Table E3-20 shows the numeric data of graphic E3-16.

	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$0.02 < \|e\| < 0.03$	$0.03 < \|e\| < 0.04$	$0.04 < \|e\| < 0.05$	$\|e\| > 0.05$
no limitation of a_{Addsub}	85.4 %	5.5 %	2.6 %	2.2 %	1.4 %	2.9 %
a_{Addsub}< 0,5	87.2 %	6.5 %	3.3 %	1.7 %	0.9 %	0.4 %
a_{Addsub}< 0,3	92.0 %	5.8 %	1.8 %	0.4 %	0 %	0 %
a_{Addsub}< 0,1	99.5 %	0.5 %	0 %	0 %	0 %	0 %

Table E3-20: Error distribution of signal Ov with different activity limitations of signal Addsub

The error of the other signals also diminish when *Addsub* activity is limited, but they have not been shown because they already had a small enough error without that limitation (lower than 0.01).

Similar results are obtained when the operand **bus width is augmented**. The error of the less-significant eight bits of the result signal (*Z*) of an adder/subtracter of more than eight bits is the same as the error of the result signal of an 8-bit adder/subtracter. The other bits of the result signal, the most-significant bits, have smaller errors than these first eight bits. For example, for a 12-bit adder/subtracter, all the errors of bits from 8 to 11 are lower than 0.01, and also for the carry signals (*Co*).

The error of the overflow signal (*Ov*) of a 12-bit adder/subtracter is similar to the error of the overflow signal of an 8-bit adder/subtracter. The maximum errors are the same as the ones shown in table E3-16 and the error distribution is slightly better for the 12-bit one.

As a conclusion, if an adequate partition is made (avoiding partitioning *Z*(0) and *Z*(1)), **none of the signals have error except for signal** *Ov*. Signal *Ov* only has error in conditions of extreme high activities. These **conditions are not usual and can be detected during the partition** process.

E3.3.1.6 Circuit "alu8051_simp"

This circuit cannot be partitioned in reconvergent regions (§E3.2.6); therefore, in order to effectively limit the BDD size, the circuit should be partitioned in disjoint regions. Performing this partition, the maximum BDD size is limited to a fixed value independent of the operand bus width (see table AIII-16).

Nevertheless, disjoint partitions provoke calculation errors. Fortunately, the errors are not too high. Table E3-21 shows the error distribution of each output port. Note that only a few of errors are higher than 0.01 and there are not errors higher than 0.02.

distribution	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$\|e\| > 0.02$
Z(7)	97.9 %	2.1 %	0 %
Z(6)	97.8 %	2.2 %	0 %
Z(5)	97.8 %	2.2 %	0 %
Z(4)	98.0 %	2.0 %	0 %
Z(3)	98.1 %	1.9 %	0 %
Z(2)	98.8 %	1.2 %	0 %
Z(1)	100 %	0 %	0 %
Z(0)	98.7 %	1.3 %	0 %
Co(1)	100 %	0 %	0 %
Co(0)	100 %	0 %	0 %
Ov	100 %	0 %	0 %

Table E3-21: Error distribution of circuit "alu8051_simp"

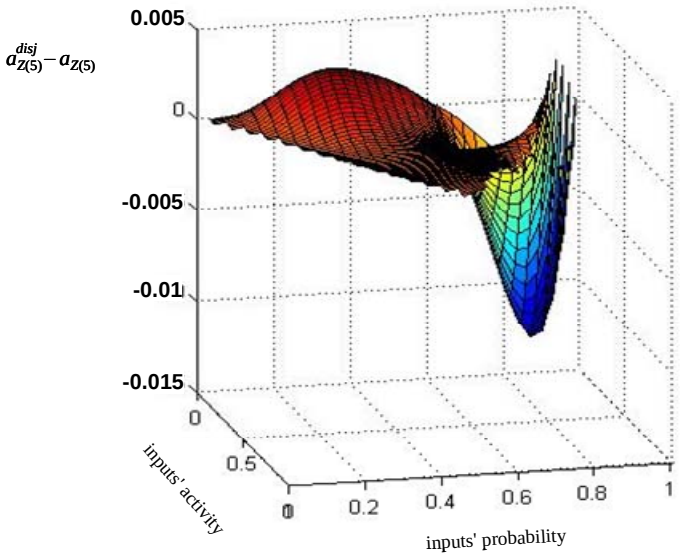
Signal *Z*(5) has the maximum negative error: $e_{max-} = -0.012$. Signal *Z*(0) produces the maximum positive error: $e_{max+} = 0.011$. These errors are low, since they are close to 0.01, what has been considered a negligible error in this Thesis.

The mean errors are much lower than 0.01. Table E3-22 shows the mean errors, standard deviation and maximum errors.

Signal	μ	σ	e_{max-}	e_{max+}
Z(7)	0.0014	0.0023	-0.012	0.003
Z(6)	0.0014	0.0024	-0.012	0.003
Z(5)	0.0014	0.0024	-0.012	0.003
Z(4)	0.0014	0.0023	-0.012	0.003
Z(3)	0.0014	0.0023	-0.012	0.003
Z(2)	0.0014	0.0021	-0.011	0.003
Z(1)	0.0011	0.0015	-0.009	0.003
Z(0)	0.0018	0.0022	-0.005	0.011
Co(1)	0.0002	0.0003	-0.0001	0.0014
Co(0)	0.0003	0.0005	-0.0003	0.0019
Ov	0.0003	0.0004	-0.0004	0.0015

Table E3-22: Mean error, standard deviation and maximum errors for each output port. Input conditions are the same for all inputs

Graphic E3-17 shows the error distribution for signal *Z*(0) for all the mesh of probability-activity pairs. As it can be observed from the graphic and from table E3-21, only a few of errors (2.2%) are higher than 0.01.



Graphic E3-17: Error distribution of signal Z(5) of circuit "alu8051_simp" for all the range of possible probabilities and activities of the inputs (same value for all the inputs)

Therefore, the error may be considered insignificant. This low error is due to the medium size partitions, that is, the circuit has not been partitioned exhaustively.

A **new analysis** has been carried out over this circuit. This analysis takes the probabilities and activities considering the operands (*A*, *B*) separately. Instead of assigning all the mesh of probability and activity pairs to the operands, the mesh is assigned to only one operand (*B*) and the other inputs, whilst the other operand (*A*) is left with constant probability and activity values.

Thus, in this experiment, input port *A* has a fixed probability and activity, and the other input ports (*B*, *Cmd*, *Ov*, *Ci*) receive all the probability and activity pairs.

In a **first experiment**, the probability and activity assigned to *A* was: $P_{A(i)} = 0.5$ and $a_{A(i)} = 0.5$. Thus, signal *A* does not have temporal correlation. The other input ports have been assigned all the probability and activity pairs. In these conditions, all the output ports in all the 1024 cases had an **absolute error lower than 0.01**.

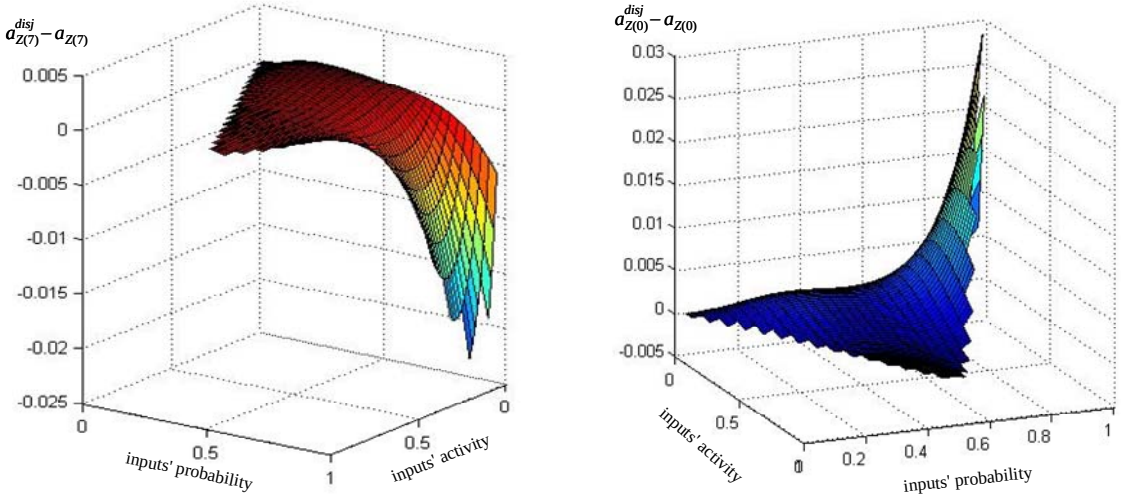
Signal *Co*(1) has the maximum positive error among all the signals, which is e_{max+} = 0.0014 (when P_i=0.43 ; a_i=0.19). Signal *Co*(1) also has the maximum negative error: e_{max-} = –0.0010 (when P_i=0.56 ; a_i=0.87). The other signals, except for *Co*(0), have errors lower than 0.0005. In consequence, it can be considered that there is **no error** in this experiment.

In the previous example, port *A* does not have temporal correlation. Thus, a **second experiment** has been accomplished in which port *A* is assigned $P_{A(i)}$ = 0.5 and $a_{A(i)}$ = 0.25. Table E3-23 shows the error distribution for each output port.

distribution	|e| < 0.01	0.01 < |e| < 0.02	|e| > 0.02
Z(7)	96.1 %	3.7 %	0.2 %
Z(6)	96.1 %	3.7 %	0.2 %
Z(5)	96.2 %	3.6 %	0.2 %
Z(4)	96.8 %	3.2 %	0 %
Z(3)	97.8 %	2.2 %	0 %
Z(2)	99.8 %	0.2 %	0 %
Z(1)	100 %	0 %	0 %
Z(0)	93.5 %	4.7 %	1.8 %
Co(1)	100 %	0 %	0 %
Co(0)	100 %	0 %	0 %
Ov	100 %	0 %	0 %

Table E3-23: Error distribution of circuit "alu8051_simp"maintaining a constant probability and activity for signal A: $P_{A(i)}$=0.5 ; $a_{A(i)}$=0.25

In the majority of the cases, the error is negligible. The worst cases are in *Z*(7) and *Z*(0), where in a 3.9% and a 6.5% of the total number of cases they were higher than 0.01, respectively. Graphic E3-18 shows the error distribution of these signals. The maximum errors occur at high input probabilities P_i = 0.94. For input probabilities lower than 0.82 for *A*(7) and lower than 0.79 for *A*(0) the errors are lower than 0.01.



Graphic E3-18: Error distribution of signals Z(7) and Z(0) of circuit "alu8051_simp" for all the range of possible probabilities and activities of the inputs, except for signal A, which remains with constant probability and activity $P_{A(i)}$=0.5 ; $a_{A(i)}$=0.25

The mean error, standard deviation and negative and positive maximum errors are shown in table E3-24.

Signal	μ	σ	e_{max-}	e_{max+}
Z(7)	0.0017	0.0031	-0.022	0.002
Z(6)	0.0017	0.0031	-0.022	0.002
Z(5)	0.0017	0.0030	-0.021	0.002
Z(4)	0.0016	0.0027	-0.019	0.002
Z(3)	0.0014	0.0023	-0.016	0.002
Z(2)	0.0011	0.0014	-0.010	0.002
Z(1)	0.0007	0.0006	-0.001	0.003
Z(0)	0.0027	0.0044	-0.002	0.029
Co(1)	0.0006	0.0005	-0.002	0.002
Co(0)	0.0003	0.0003	-0.0003	0.001
Ov	0.0002	0.0002	-0.0001	0.001

Table E3-24: Mean error, standard deviation and maximum errors for each output port. Input conditions are the same for all inputs except for signal A, which remains with constant probability and activity $P_{A(i)}=0.5$; $a_{A(i)}=0.25$

The results of this analysis may be considered valid, only in some special few cases the error increment; nevertheless, in those cases the error is not too high (lower than 0.03).

As a conclusion, these experiments prove the validity of the estimations resulting from the disjoint partition. The results have been proved even when the probabilities and activities have been grouped considering the operands.

E3.3.1.7 Circuit "alu8051"

Section E3.2.7 showed the BDD sizes resulting from two disjoint partitions: exhaustive and minimum partitions. In the exhaustive disjoint partition of the 8-bit circuit, 596 regions were obtained. The largest *a*BDDs of these regions only had 13 nodes. It is obvious that it is not necessary to obtain such small areas. In contrast, the largest *a*BDDs of the minimum disjoint partition had 912 nodes, which is still a manageable amount and even larger number of nodes could be handled.

One of the advantages of **disjoint partitions** is that usually the largest BDD size **remains constant with the bus width**.

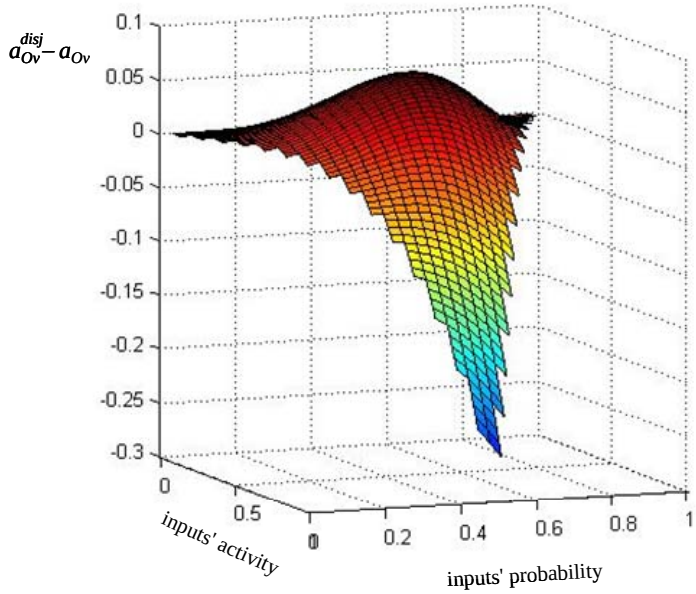
Table E3-25 shows the error distribution of all output ports when the exhaustive partition is applied.

signal	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$0.02 < \|e\| < 0.03$	$0.03 < \|e\| < 0.04$	$0.04 < \|e\| < 0.05$	$\|e\| > 0.06$
Z(7)	97.8 %	2.2 %	0 %	0 %	0 %	0 %
Z(6)	99.1 %	0.9 %	0 %	0 %	0 %	0 %
Z(5)	100 %	0 %	0 %	0 %	0 %	0 %
Z(4)	100 %	0 %	0 %	0 %	0 %	0 %
Z(3)	100 %	0 %	0 %	0 %	0 %	0 %
Z(2)	100 %	0 %	0 %	0 %	0 %	0 %
Z(1)	100 %	0 %	0 %	0 %	0 %	0 %
Z(0)	100 %	0 %	0 %	0 %	0 %	0 %
Co(1)	63.9 %	24.8 %	11.3 %	0 %	0 %	0 %
Co(0)	53.5 %	19.3 %	14.0 %	11.2 %	2.0 %	0 %
Ov	29.6 %	15.1 %	13. 3 %	11.0 %	9.8 %	21.2 %

Table E3-25: Error distribution of the exhaustive partition of circuit "alu8051"

Signal *Z* has not a significant error; however, signals *Ov* and *Co* have higher errors, especially *Ov*, whose maximum error is -0.29. Although this error happens in extreme conditions: P_{in}=0,5 and a_{in}=1, less than the 30% of the cases have an error lower than 0.01.

The error distribution of signal *Ov* has been plotted in graphic E3-19.



Graphic E3-19: Error distribution of signal Ov of circuit "alu8051" being applied the exhaustive partition. All the inputs have been assigned the mesh of probability-activity pairs

As it has been said, in order to minimize the error, the activity of the selection signal should be low. That is one of the reasons why the error is high in the area close to *a*=1.

The selection signal of this circuit is 6 bit width (*Cmd*); therefore, unless each bit has a low activity, the global activity of the signal will not decrease significantly.

The global activity of *Cmd* is:

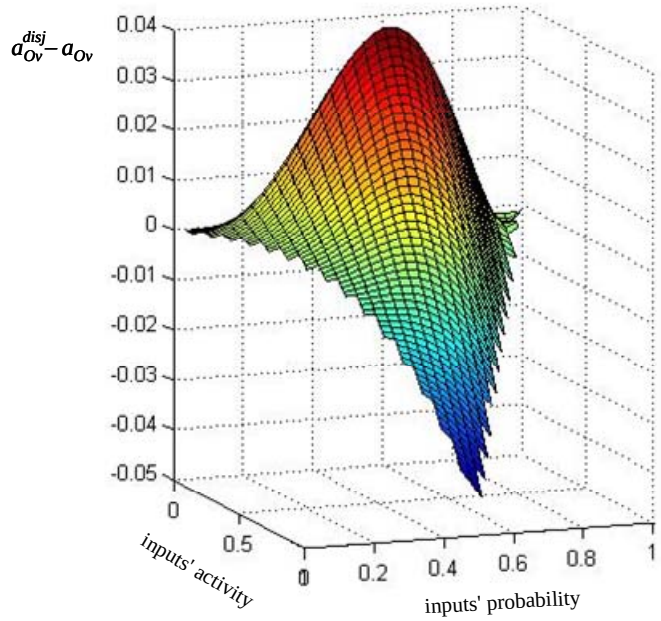
$$a_{Cmd} = a_{Cmd(0)} + a_{Cmd(1)}\cdot\overline{a}_{Cmd(0)} + a_{Cmd(2)}\cdot\overline{a}_{Cmd(1)}\cdot\overline{a}_{Cmd(0)} + a_{Cmd(3)}\cdot\overline{a}_{Cmd(2)}\cdot\overline{a}_{Cmd(1)}\cdot\overline{a}_{Cmd(0)} +$$
$$+ a_{Cmd(4)}\cdot\overline{a}_{Cmd(3)}\cdot\overline{a}_{Cmd(2)}\cdot\overline{a}_{Cmd(1)}\cdot\overline{a}_{Cmd(0)} + a_{Cmd(5)}\cdot\overline{a}_{Cmd(4)}\cdot\overline{a}_{Cmd(3)}\cdot\overline{a}_{Cmd(2)}\cdot\overline{a}_{Cmd(1)}\cdot\overline{a}_{Cmd(0)}$$

Thus, if each bit activity is limited to 0.1, the global activity of *Cmd* will be 0.47 at most. Table E3-26 shows the error distribution under these conditions.

distribution	$\|e\| < 0.01$	$0.01 < \|e\| < 0.02$	$0.02 < \|e\| < 0.03$	$0.03 < \|e\| < 0.04$	$0.04 < \|e\| < 0.05$	$\|e\| > 0.06$
Z(7)	98.7 %	1.3 %	0 %	0 %	0 %	0 %
Z(6)	99.8 %	0.2 %	0 %	0 %	0 %	0 %
Z(5)	100 %	0 %	0 %	0 %	0 %	0 %
Z(4)	100 %	0 %	0 %	0 %	0 %	0 %
Z(3)	100 %	0 %	0 %	0 %	0 %	0 %
Z(2)	100 %	0 %	0 %	0 %	0 %	0 %
Z(1)	100 %	0 %	0 %	0 %	0 %	0 %
Z(0)	100 %	0 %	0 %	0 %	0 %	0 %
Co(1)	71.4 %	26.4 %	2.2 %	0 %	0 %	0 %
Co(0)	60.6 %	23.9 %	13.2 %	2.3 %	0 %	0 %
Ov	48.6 %	25.2 %	16.7 %	8.6 %	0.9 %	0 %

Table E3-26: Error distribution of the exhaustive partition of circuit "alu8051" when Cmd activity is limited to $a_{Cmd(i)} < 0.1$

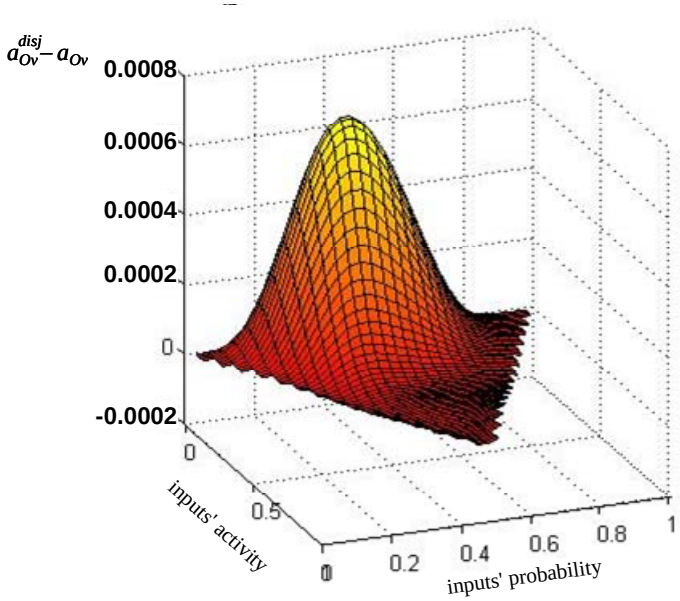
Under these conditions, a better error distribution is obtained for all signals. Nevertheless, signal *Ov* still has significant errors. Again, the worst case happens when P_{in}=0.5 and a_{in}=1, in which error is -0.046. Graphic E3-20 plots the error distribution of signal *Ov*.



Graphic E3-20: Error distribution of signal Ov of circuit "alu8051" being applied the exhaustive partition. All the inputs have been assigned the mesh of probability-activity pairs, except for signal Cmd, whose bits activities have been limited: $a_{Cmd(i)} < 0.1$

The error produced when the **minimum disjoint partition** is applied is totally different. Under this partition, **all the errors remain below 0.01**. Therefore, it may be considered that **no error is produced** even though the selection signal activity has not been limited.

The error distribution of signal *Ov* has been plotted in graphic E3-21. The errors are some orders of magnitude lower than those of the exhaustive partition.



Graphic E3-21: Error distribution of signal Ov of circuit "alu8051" being applied the minimum partition. All the inputs have been assigned the mesh of probability-activity pairs

Actually, what has been called the minimum disjoint partition in this experiment is not an absolute minimum partition; still, the errors are negligible because the resulting regions are

medium size. The results of these experiments **confirm that exhaustive partitions should be avoided**.

E3.4 Global Analysis

Since the results have been given circuit by circuit, in order to summarize the experiments, this section will give a global vision or the results. The following global analyses will be given:

- The activity BDD size reduction due to the proposed activity BDDs (*a*BDD against TFBDD)
- The *a*BDD size reduction due to the RTL analysis instead of the gate level analysis
- The *a*BDD size reduction due to the RTL disjoint partition instead of the exact RTL analysis
- The *a*BDD size reduction due to the RTL disjoint partition instead of the gate level analysis
- The error caused by the RTL disjoint partition when it implies an advantage

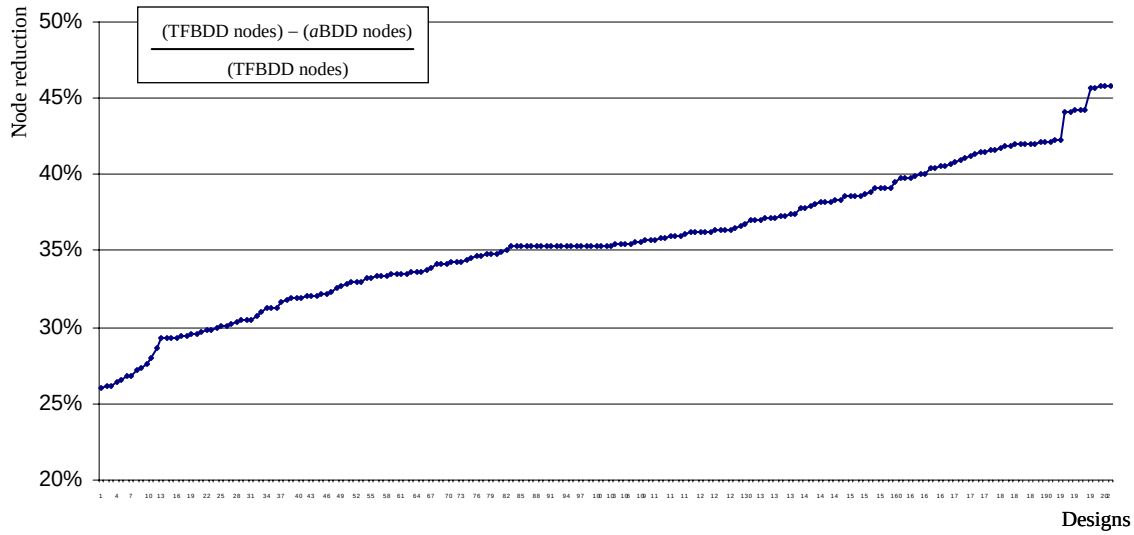
The following analyses intend to provide a simplified perspective of the benefits; therefore, BDD comparisons will be related mainly to *a*BDDs. Probability BDDs are much smaller than activity BDDs; thus, they are not that critical. In the other hand, TFBDDs usually are about a 30% larger than *a*BDDs; therefore, comparisons between TFBDDs are similar to those between *a*BDDs.

All the data given in the next subsections is extended in the individual analysis.

E3.4.1 Node reduction due to the *a*BDDs

The seven circuits used have been analyzed at gate level, at RTL, and at RTL performing disjoint partitions. Some of the circuits have been modified changing their operand bus width. Besides, some of the circuits have been synthesized in various gate level circuits and their gate arrangement have been modified (for example, circuit "max" in annex AIII.1.1, in Spanish). As a result, more than 200 analyses have been performed and in all the cases a reduction in BDD size has been achieved by the usage of *a*BDDs instead of TFBDDs.

Graphic E3-22 shows the node reduction obtained by the usage of *a*BDDs. For each design, the number of nodes of all the *a*BDDs of the circuit has been compared with the number of nodes of all TFBDDs of the same circuit. The graphic X-axis has been ordered according to the node reduction achieved.



Graphic E3-22: Node reduction obtained by the usage of aBDDs instead of TFBDDs

The leftmost circuit is the RTL 15-bit adder/subtracter, in which the *a*BDD achieves a 26% reduction in the number of nodes. The rightmost circuit is the disjoint-RTL ALU, in which the *a*BDD achieves a 46% reduction in the number of nodes.

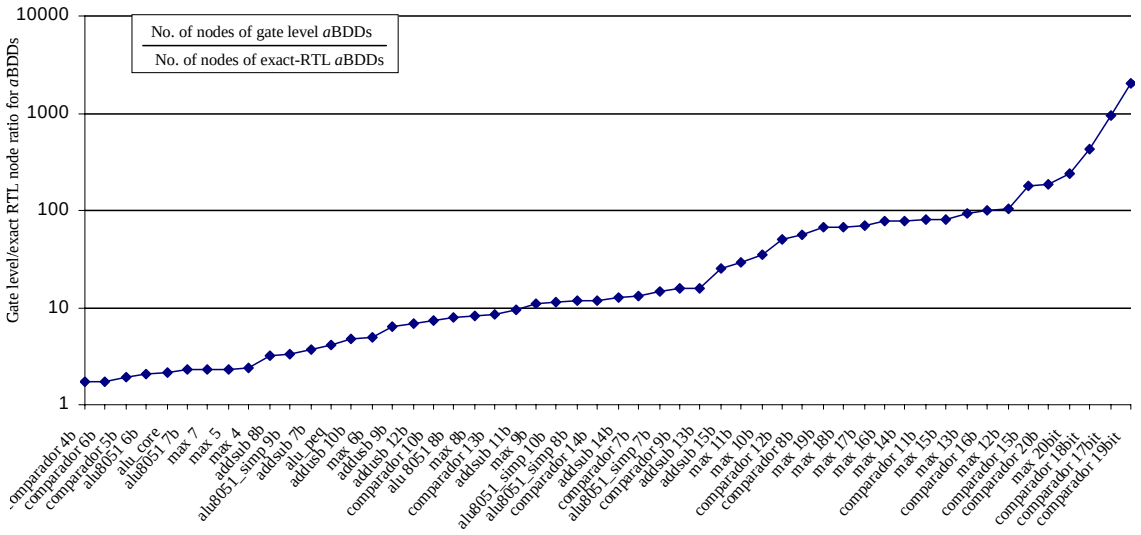
Therefore, **the usage of the proposed activity BDDs attains a node reduction between 26% and 46%**. The proposed activity BDDs (*a*BDD) do not produce any inaccuracy in the calculations.

E3.4.2 BDD node reduction due to the analysis at RTL against gate level

The RTL analysis without disjoint partition usually produces smaller BDDs than the gate level BDDs. Since this RTL analysis does not partition the circuit in disjoint regions, the analysis does not produce error, and that is the reason why it has been called exact-RTL analysis.

The exact-RTL analysis produces smaller BDDs due to its better ordering and also because gate level synthesis sometimes increments circuit reconvergences.

Graphic E3-23 shows the ratio of the number of nodes of the gate level *a*BDDs to the number of nodes of the exact-RTL *a*BDDs for each circuit. Note that the scale is logarithmic. Note also that the differences typically increase when the operand bus width increases too.



Graphic E3-23: Ratio of gate level aBDD nodes to exact-RTL aBDD nodes

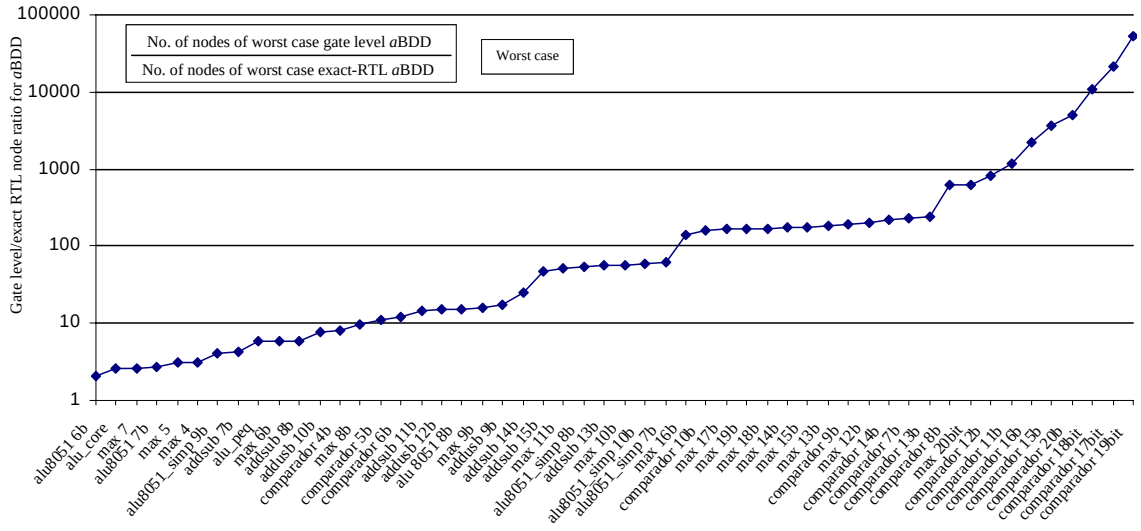
The four-bit "comparator" (circuit 2) has the smaller ratio: 1.74; while the same circuit but having 19-bit width has the higher ratio: the gate level circuit has more than 2000 times the number of nodes of the exact-RTL circuit. These data also confirms the influence of the bus width on the ratio. That is, the larger the circuit (the wider the bus) is, the better the advantages of the RTL analysis.

Nevertheless, the most interesting results are those that could not appear in the graphic because of the excessive memory resources needed for the gate level analysis. On the contrary, these analyses were satisfactorily performed at RTL. These analyses are:

- Circuit "max" (circuit 1): it was not possible to perform the analysis of the 21-bit bus width circuit or wider. Even some of the circuit 20-bit bus width versions also exceeded the memory computer resources.
- Circuit "comparador" (circuit 2): it was not possible to analyze the circuit from the 20-bit bus width circuit onwards

- The simplified 8051 ALU ("alu8051_simp", circuit 6): it was not possible to analyze the circuit from the 11-bit bus width circuit onwards
- The 8051 ALU ("alu8051", circuit 7): it was not possible to analyze the circuit from the 9-bit bus width circuit onwards.

The comparison between the worst case exact-RTL *a*BDDs and the worst case gate level *a*BDDs is also favorable to the exact-RTL analysis. The worst case *a*BDD is very important because it may determine whether the analysis could be made or not. Graphic E3-24 shows the ratio of the worst case gate level *a*BDD to the worst case exact-RTL *a*BDD. Note that the ratios are higher than those of the previous graphic, where the number of nodes of all circuit *a*BDDs were accounted. Notice that the differences typically increase when the operand bus width increases too.

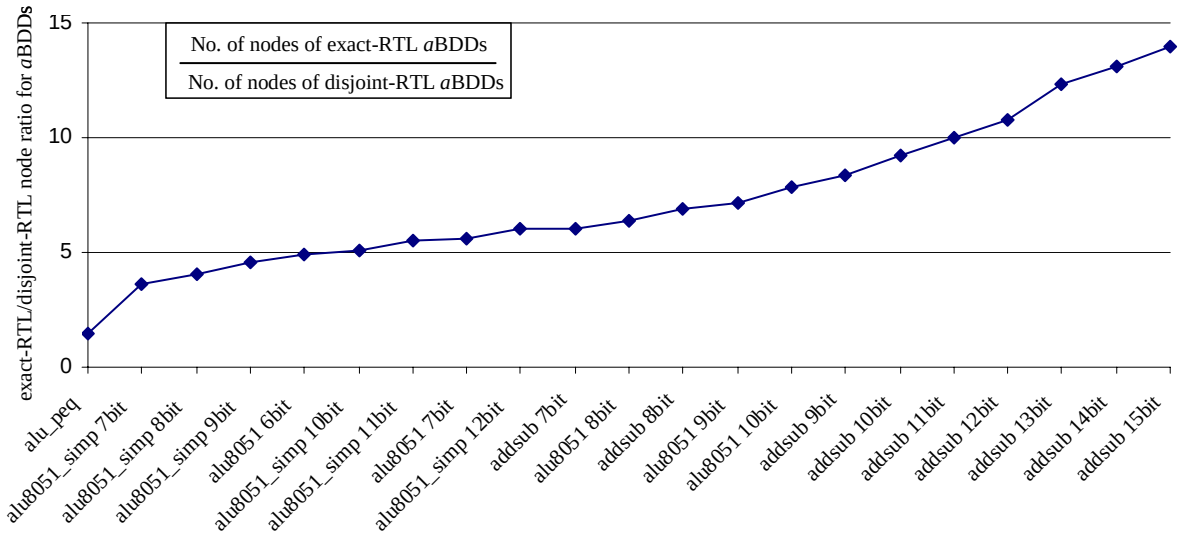


Graphic E3-24: Node ratio of worst case gate level aBDD to worst case exact-RTL aBDD

E3.4.3 BDD node reduction due to the disjoint-RTL analysis against exact-RTL analysis

As it has been stated, the disjoint partition should only be made with large BDDs and when the circuit conditions recommend performing it. These circuit conditions can be checked during the RTL analysis using the extended hardware model (EHM §3.3.2).

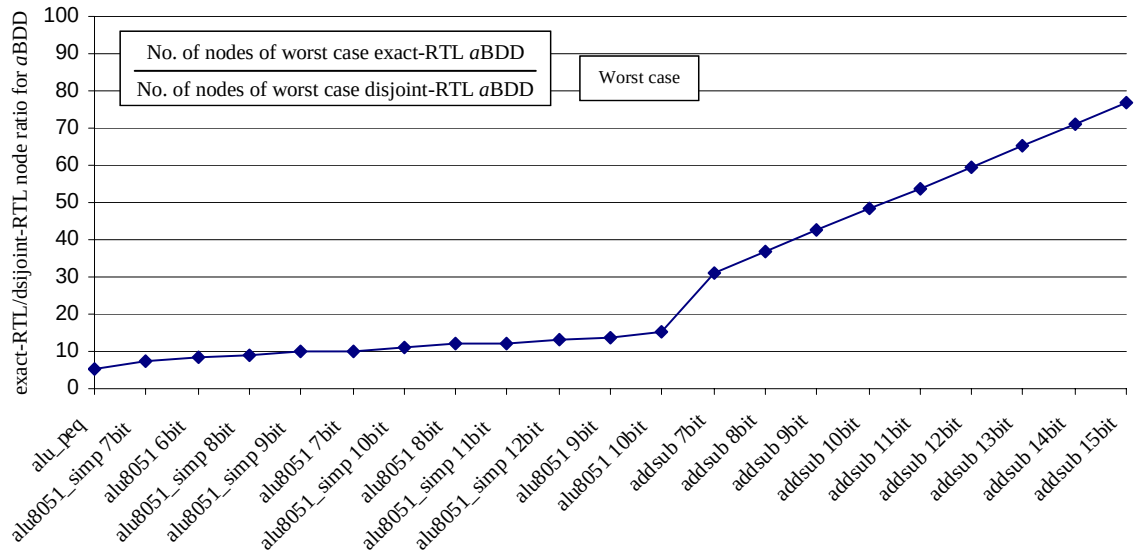
Graphic E3-25 shows the ratio of the number of nodes of the exact-RTL *a*BDDs to the number of nodes of the disjoint-RTL *a*BDDs for each circuit. Note that the scale is logarithmic. Note also that the differences typically increase when the operand bus width increases too. The exhaustive disjoint partitions have not been included because they are not advisable.



Graphic E3-25: Ratio of exact-RTL aBDD nodes to disjoint-RTL aBDD nodes

It can be observed from graphic E3-25 that the larger the bus width is the higher the ratio is. The smallest ratio is for circuit "alu_peq", which is only 1.5. Circuit "alu_peq" (§4.2.3) is a small circuit; thus, it is not really necessary to partition it.

The comparison between the disjoint-RTL worst case *a*BDDs and the exact-RTL worst case *a*BDDs is also favorable to the disjoint-RTL analysis. Besides, the disjoint-RTL worst case *a*BDDs do not grow with the bus width. Graphic E3-26 shows the ratio of the worst case exact-RTL *a*BDD to the worst case disjoint-RTL *a*BDD.



Graphic E3-26: Node ratio of worst case exact-RTL aBDD to worst case disjoint-RTL aBDD

It has been observed that, in those circuits where the operand bus width has been modified, the worst case disjoint-RTL *a*BDDs remain constant in size; whilst the size of the worst case exact-RTL *a*BDDs increments linearly with the bus width.

Next, in table E3-27 shows the size of the largest (worst case) *a*BDDs for 8-bit and *n*-bit circuits depending on whether the disjoint partition has been made.

Circuit	exact-RTL		disjoint-RTL	
	No. of nodes of the worst case *a*BDD for 8 bits	No. of nodes of the worst case *a*BDD for *n* bits	No. of nodes of the worst case *a*BDD for 8 bits	No. of nodes of the worst case *a*BDD for *n* bits
max[78]	110	$15 \cdot n - 10$	19	19
comparador	12	12	12	12
addsub	479	$74 \cdot n - 113$	13	13
alu8051_simp	3215	$423 \cdot n - 169$	370	370
alu8051	10807	$1467 \cdot n - 929$	912	912

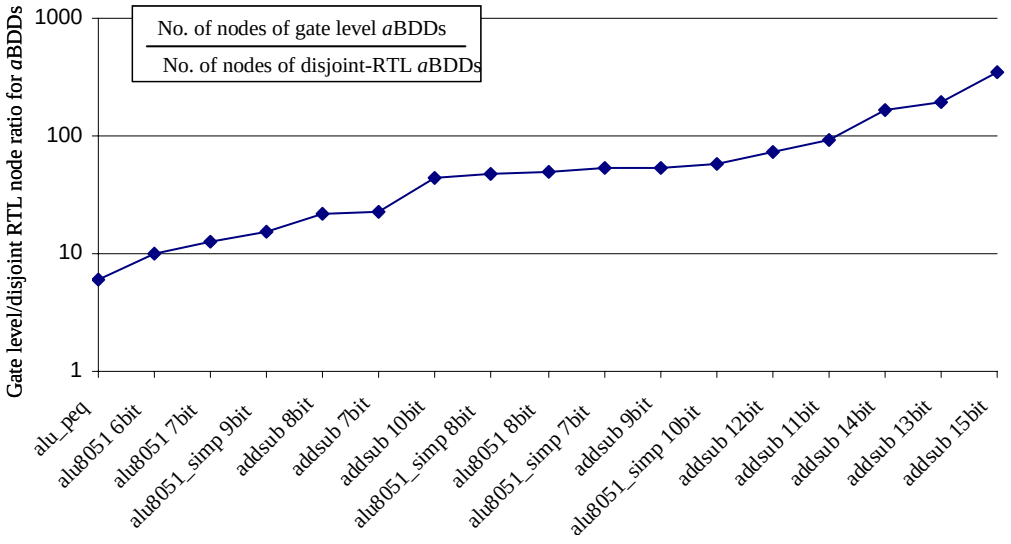
Table E3-27: Worst case aBDD sizes for 8-bit and n-bit bus width circuits depending on whether the disjoint partition has been made

It can be observed from table E3-27 that the more complex the circuit is, the higher the growth of the worst-case exact-RTL is.

E3.4.4 BDD node reduction due to the disjoint-RTL analysis against gate level analysis

The previous sections showed that the disjoint-RTL analysis usually produces smaller BDDs than the exact-RTL analysis. In addition, the exact-RTL analysis produces smaller BDDs than the gate level analysis. Therefore, larger differences between disjoint-RTL analysis and gate level analysis should be expected. The following graphics support this reasoning.

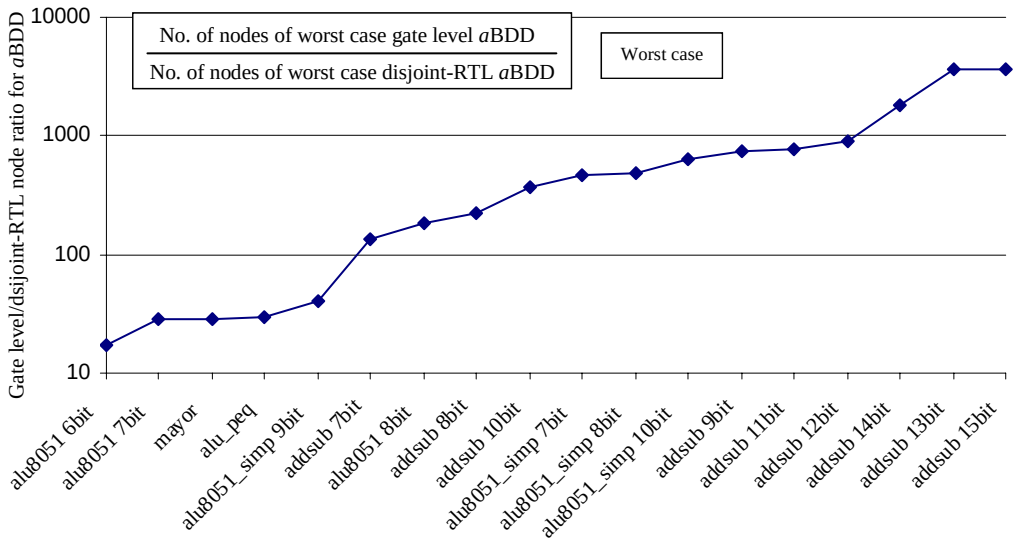
For each circuit, graphic E3-27 shows the ratio of the number of nodes of the gate level *a*BDDs to the number of nodes of the disjoint-RTL *a*BDDs. Note that the scale is logarithmic.



Graphic E3-27: Ratio of gate level aBDD nodes to disjoint-RTL aBDD nodes

Graphic E3-28 shows the ratio of the worst case gate level *a*BDD to the worst case disjoint-RTL *a*BDD.

[78] Circuit "max" has been included even though it has not been partitioned in minimum disjoint regions



Graphic E3-28: Node ratio of worst case gate level aBDD to worst case disjoint-RTL aBDD

These graphics evidence the advantages related to the BDD size of the disjoint-RTL analysis against the gate level analysis.

E3.4.5 Error caused by the RTL disjoint partition

The proposed method has two modes depending on whether the disjoint partition is made. When the circuit is not partitioned in disjoint regions the method is exact. In all the RTL experiments, the analysis could be done without resorting to disjoint partitioning. That was not possible at gate level, where some circuits could not be analyzed due to their large BDD sizes. Therefore, the proposed method permits to **calculate the activity of the circuit analyzed without error**. Nevertheless, the disjoint partition provides a significant node reduction in BDD size and it can be expected that, even at RTL, for large circuits, disjoint partitioning is necessary in order to avoid too large BDDs.

This section shows the error produced in each circuit by the minimum disjoint partition. The error caused by exhaustive disjoint partitions will not be shown because these partitions are not advisable. In previous sections some results of exhaustive disjoint partitions have bee shown in order to demonstrate that it is not recommended, but not as a partition proposal.

For each circuit, table E3-28 shows the error statistics caused by the disjoint partitions. The maximum errors ($|e_{max}|$) are in absolute values. The smallest circuits ("max", "comparador" and "alu_core") have not been partitioned due to their small BDDs; thus, those circuits have no error.

Circuit		Worst case *a*BDD		Error statistics			Commentaries
No.	name	exact-RTL	disjoint-RTL	μ	σ	$\lvert e_{max} \rvert$	
1	max	50	-	0	0	0	no disjoint partition
2	comparador	12	-	0	0	0	no disjoint partition
3	alu_peq	438	85	0.0080	0.0143	0.109	
4	alu_core	168	-	0	0	0	no disjoint partition
5	addsub	479	13	0.0011	0.0049	0.125	
6	alu8051_simp	3215	370	0.0011	0.0019	0.012	
7	alu8051	10807	912	0.00056	0.00088	0.0068	

Table E3-28: aBDD node reduction and error statistics when partitioning the circuit in disjoint regions

It can be observed from the table that, the smaller the exact-RTL *a*BDDs are, the higher the errors are. Circuits "alu_peq" and "addsub" have the highest errors and their exact-RTL *a*BDDs

have 438 and 479 nodes, respectively; whilst circuit "alu8051_simp", having an exact-RTL *a*BDD with 3215 nodes, has lower errors. Observe that the mean error of "alu8051_simp" is equal to the "addsub" error; however, its standard deviation and maximum error is much lower. Last, circuit "alu8051" has negligible errors, even its maximum error.

These results may suggest that circuits "alu_peq" and "addsub" are too small to perform the disjoint partition. Even for "alu8051_simp" the partition could not be done, and only for circuits similar to "alu8051" or larger the partition should be made. This decision can be taken during the circuit analysis because the EHM provides information to estimate the resulting BDD size.

Nevertheless, even though circuits "alu_peq" and "addsub" have some important errors, many of these errors are provoked by extreme probability and activity input conditions. These conditions are improbable during the normal circuit operation, but these conditions are the ones that worsen the error statistics. As it has been explained in previous sections (§E3.3.1), during the error analyses some experiment modifications were carried out to avoid these improbable conditions. Annex table AIII-18 (in Spanish) enlarge the information of table E3-28, where those experiments are included. As it can be seen, those modifications significantly reduce the error.

This last remark implies that for circuit "alu8051", despite of those improbable conditions, even the maximum error remains negligible.

To end, table E3-28, or even better table AIII-18, may be compared with table E3-9, in which the errors of other proposals are shown. As it was said in section E3.3.1, the results are not directly comparable, but at least they may be used as a reference.

E3.5 Processing times

The analyses have been carried out through a computer tool whose characteristics are explained in more detail in annex IV (in Spanish). The tool is still in development status; thus, it is not optimized yet. It is a large, complex program because it creates the EHM, analyzes the reconvergences, partitions the circuit in disjoint regions if advisable, builds the BDDs, computes the activities and propagates them, amongst other functions. Many revisions and a lot of debugging have been necessary to create the program. The program has fallen into processing redundancies that made the estimation procedure excessively slow. At present, the code is much more efficient, but it still needs to be debugged and optimized. Therefore, the processing times are not reliable yet.

For the analyses, an Intel Pentium 4 3.20 GHz computer with 1 GB RAM has been used. The final program is executed using Cygwin [32], which is a Linux-like environment for Windows. Therefore, if similar computers are used, programs running on Cygwin will be slower than those programs running in Linux. Besides, the time measure given by Cygwin is not exact (using the Cygwin command *time*). The same program execution may give time differences by over 50%.

Table E3-29 shows the processing times for circuit "alu8051" (circuit 7), which is the most complex circuit. The other circuit processing times are lower. Circuits "alu8051" and "alu8051_simp" are the only circuits that their processing times at RTL level take more than a second. The other circuits and for any other bus width that has been analyzed, the processing times take less than a second.

time (s)	Gate level	exact-RTL	disjoint-RTL
6 bits	830 (13' 50")	4	3,5
7 bits	7494 (2h 4' 54")	5	4
8 bits	21536 (5h 58' 56")	6	4,5
9 bits	*excesive memory required*	7	5
10 bits	*excesive memory required*	8	5,5

Table E3-29: Processing times in seconds for circuit "alu8051" (circuit 7) depending on the bus width and the method of analysis

The data from table E3-29 should be taken cautiously because the program need to be further optimized. The code has been constantly debugged. At the beginning, the processing times were some orders of magnitude higher than those of table E3-29: the gate level analysis took more than a week and the RTL analysis took hours. At present time, it has been detected that some parts of the code need to be optimized. These code optimizations would probably reduce the processing times in great manner. However, they have not been carried out because, at this moment, the processing times are acceptable at RTL and therefore, the **proposed method is feasible**.

The program could have been optimized in order to demonstrate the processing time differences between gate level and RTL. Nevertheless, these optimizations imply a high developing cost due to the program size (some tens of thousands of code lines). This program improvement is not essential to demonstrate the benefits of the RTL analysis against the gate level analysis. The experimental results that compare the BDD sizes may be considered enough to prove the advantages of the RTL analysis.

Besides, a comparison between the probabilistic methods (used in this Thesis) and the simulative methods has been obtained. The first two circuits (the 4-bit versions of circuits "max" and "comparador") have been taken for the analysis. Zero-delay simulations have been carried out using Modelsim [85], in which a program has included to monitor signals' values and transitions. Using 16384 (2^{14}) test vectors, the simulation took more than 4 minutes for the RTL circuit; whilst the same simulation for the gate level circuit took almost 10 minutes. The probabilistic analysis of these circuits took 0.2 seconds. For these circuits, due to their small BDD sizes, the probabilistic analyses took the same time regardless of whether the circuit is analyzed at gate level or RTL. Therefore, these examples confirm the advantages of the probabilistic methods against the simulative. These advantages increase with the circuit size because simulation times are larger.

Last, the probabilistic method is a one-time processing, which is one of its main advantages. Once the probabilistic model has been built, if the input conditions change, it is not necessary to rebuild the model, just the input activities and probabilities are propagated. Hence, the processing times are much lower.

E3.6 Conclusions of the experimental results

The present chapter not only has allowed the demonstration of the advantages of the activity analysis at RTL against the analysis at gate level but also has been useful to prove the benefits of some of the particular proposals of this Thesis.

The results presented have proved that **the RTL analysis is more convenient than the analysis at gate level**. The RTL analysis without disjoint partitions (exact-RTL) does **not produce any inaccuracies** and provides the following advantages:

- **The RTL analysis provides an adequate ordering**. Contrary to what happens at gate level, in which the orderings are unpredictable. The graphics of section E3.4.2 compare the BDD sizes and demonstrate the validity of the ordering proposal (§E2.6). In most of the cases, the sum of the nodes of all activity BDDs are more than the double for the gate level analysis

than for the RTL analysis. In some of the cases, this sum is more than a thousand times larger in the gate level analysis. The RTL ordering is obtained by the analysis of the RTL structures and operators, and also by the knowledge of the indexes of multi-bit operands.

- These **differences rise when the circuit size and the number of inputs increase**. If the bus width is incremented, the **gate level** BDDs increment in such a way that it is **not possible to analyze** the circuit. Whereas those circuits can be analyzed at RTL.
- When the circuits' bus width was incremented in the circuits being analyzed, the worst-case exact-RTL BDDs increased their size linearly. While the increment of the worst-case gate level BDDs was unpredictable and showed a higher tendency than linear.
- Sometimes **logic synthesis increments circuit reconvergences**. As a result, the gate level analysis has to handle larger partitions. Reconvergent partitions do not produce inaccuracies; therefore, it is an advantage of RTL analysis because smaller BDDs are obtained with no exactitude cost.
- The **RTL analysis allows to perform the disjoint partition in an easy way**. On the other hand, at gate level, finding the disjoint regions is complicated. The results of the partition are not exact, but the error is minimized when the regions are kept moderately large and the proposed conditions for the partition are observed.
- The RT level facilitates the comprehension of the circuit functionality, allowing an analysis that makes possible performing the partitions taking into account the circuit functionality (annexes AIII.5.1 and AIII.7.1 in Spanish)

In addition to these advantages, the benefits of the particular proposals have also been proved:

- The **proposed activity BDD representation** (*a*BDD §E2.5.2) **always produces smaller BDDs**. Section E3.4.1 showed that the number of nodes were reduced from 26% to 46%. These experiments corroborate the theoretical studies expounded in section E2.5.
- For large circuits, the **disjoint-RTL partition produces smaller BDDs** than those coming from exact-RTL analysis (§E3.4.3) or those coming from the gate level analysis (E3.4.4).
- The BDD size differences increment with the circuit size and the circuit number of inputs. Therefore, **disjoint-RTL partitions are more favorable for large circuits**. As show in the experiments, worst case disjoint-RTL BDD sizes remain constant when the bus width is incremented; whilst the worst case exact-RTL BDD sizes usually increment linearly. On the other hand, at gate level the BDD size increments are unpredictable.
- The **extended hardware model** (EHM §E2.3.2) provides an efficient way to identify the disjoint regions and analyze whether the conditions that minimizes the error are fulfilled (§E2.4.3). In order to reduce the error, exhaustive disjoint partition should be avoided but **minimum disjoint partitions** should be made.
- The error provoked by disjoint partitions can be foreseen depending on the circuit structure, the partition made and the probability and activity conditions of the involved signals. Thus, there is a **structural characteristic** and an **extrinsic characteristic**. The section devoted to the error analysis (§E3.3) showed that the error can be minimized. This section also showed that the error depends on the conditions exposed in section E2.4.3 (also in annex section AII.2, in Spanish).
- If the proposed conditions for the disjoint partition are observed and there are no extreme probability and activity conditions, the error is kept in acceptable values (below 0.01).
- As it has been said (§E3.1), the computer program limitations has restricted the size of the circuits to be analyzed. Nevertheless, the experiments lend support to the assumption that the minimum disjoint partition is of great importance for large, complex circuits. In those circuits, a minimum disjoint partition would significantly reduce the BDD size and, at the same time, it would keep the error low, as it happened with circuit "alu8051" (circuit 7). For large circuits, it may happen that even the exact-RTL analysis has difficulties and thus, the disjoint partition could be necessary.

To conclude, this chapter has provided evidence that the **activity analysis at RTL is more advantageous than the analysis at gate level**. The RTL analysis works with smaller BDDs without losing accuracy. The BDD size differences are due to the better RTL ordering that has been proposed in this Thesis. RTL BDDs may be more than a thousand times smaller than gate level BDDs. And what is even more important, the gate level BDDs may be so large that the gate level analysis could become unfeasible; whilst the circuit can be analyzed at RTL.

This fact provokes the proposed method to be **the only one being able to perform an exact analysis of the switching activity** of medium size circuits. Further experiments should to be done for larger circuits.

In addition, for large, complex circuits the RTL BDDs can be further reduced if disjoint partitions are made. These partitions can be performed following the proposed partition conditions, and thus, limiting the error.

E4. CONCLUSIONS

This Thesis addresses the probabilistic estimation of switching activity of digital electronic circuits. The activity is a fundamental parameter for the evaluation of digital circuit dynamic power consumption. Early estimation of a circuit's activity may help reducing design flow iterations. Design flow iterations increment costs and provoke delays in market entry.

Besides, the activity probabilistic analysis may be useful for other studies such as testability, reliability, verification and behavioral analysis of the circuit.

This Thesis's principal proposal is **to raise the activity estimation abstraction level from gate level to RTL**. Two objectives are achieved with this proposal: the first one is a methodological goal and the second one is a practical goal. These two objectives are:

- To anticipate the information offered to the designer
- To take advantage of RTL characteristics in order to optimize the estimation methods

The first objective is to offer the information beforehand, what will **help the designer making correct design decisions**. The sooner these correct decisions are taken, the **fewer design iterations** are made. Therefore, this is a **methodological** objective.

The second objective is practical. This objective intends to make use of the advantages of carrying out the estimation at a higher abstraction level in order to reduce the analysis complexity. The main RTL characteristics that can be taken advantage of to optimize the estimation method are:

- The **higher simplicity** of the RTL circuit descriptions. The higher the abstraction level is, the less detailed information exists. Although this information may be useful, it increases the analysis complexity
- The **greater the high level information** of RTL circuits. RTL provides information related to the circuit operation, organization and behavior, which is hard to obtain at gate level. Using this information enormously facilitates the estimation. Examples of this kind of information are:
    - The existence of logic and arithmetic operators
    - The definition of structures such as multiplexers
    - The definition of arrays, integer types and enumerations
- **Less reconvergences**: in some cases, circuit reconvergences may increase after the logic synthesis. This increment makes the number of reconvergent regions to diminish; thus, the region sizes increment, as well as the BDD sizes. Since reconvergent partitions do not produce any inaccuracy, it is a clear advantage of the RTL analysis against the gate level analysis.

Therefore, **this Thesis proposes a probabilistic method to estimate the switching activity at RTL** that takes advantage of the RTL benefits. Conceptually, the proposed method is an extension of the existing gate-level probabilistic methods. Therefore, gate level circuits can also be analyzed with the proposed method, although without the RTL advantages. On the other hand, the raise of the abstraction level also implies some difficulties. All this has impelled to make new proposals in order to take advantage of the RTL level and overcome the problems. Besides, contributions valid for both gate level and RTL have also been proposed. All these contributions are:

- An **extended hardware model** (EHM) has been proposed so as to provide a structure where the probabilistic estimation can be performed (§E2.3). This model allows to have a similar structure to gate level netlists, including a structure of signals' dependences and influences. However, this model keeps the RTL information and simplicity. Besides, it facilitates circuit partitioning and the BDD construction, by which signal activities and probabilities are

calculated. Such activities and probabilities are propagated from inputs to outputs through the model.

- An **efficient BDD ordering based on the RTL structure** has been proposed. The BDD size is strongly dependent on the variable order and it is not easy to obtain an adequate order at gate level. However, as it has been observed in this Thesis, following the RTL arrangement provides with a good variable ordering. Moreover, using the knowledge provided by the RTL descriptions about vector indexes and operators, a rather favorable ordering is obtained.
- Performing **disjoint partitions** has been proposed for circuits whose size and complexity hinder the probabilistic analysis. An easy way to perform such partitions at RTL has been proposed. On the other hand, disjoint partitions are difficult to perform at gate level. Although disjoint partitions produce errors in activity computations, a method to perform the partition that minimizes the errors has also been proposed in this Thesis.
- A new **activity BDD representation** (*a*BDD) has been proposed. Such representation reduces the number of BDD nodes between 25% to 50% with no error. This representation is valid for both gate level and RTL. The operations for creating and manipulating the *a*BDDs have been defined in this Thesis. In addition, the methods to transform the former activity BDDs (TFBDD) into *a*BDDs have been defined.

All these proposals have been implemented in a computer tool; therefore, demonstrating their feasibility and also the validity of this Thesis' main proposal: the activity estimation at RTL.

The experimental results show that in many cases the estimation at RTL is not feasible; whilst the estimation is viable at RTL by means of the Thesis' proposals. For medium size circuits, in which the analyses can be performed at both gate level and RTL, RTL analyses have shown evident advantages over gate level analyses. Such advantages, which may be measured in various orders of magnitude, are due to the better RTL ordering and the possibility of making more partitions, both reconvergent and disjoint.

Although the same circuits of other proposals could not be taken, analyses of circuits with similar size and characteristics have shown that the proposed method is **the only one that has been able to perform an exact analysis of the activity** of medium size circuits.

Regarding disjoint partitions, if the proposed partition recommendations are followed, in most of the cases, the activity absolute error remains below 0.01. The proposal includes analyses that can be performed during the partition process and allow the evaluation of the impact of the partition over the error. Therefore, it can be assessed whether the partition should be performed and where in the circuit is more adequate to perform the partition.

In conclusion, the proposed activity estimation method provides **an efficient mean to perform the estimation beforehand** in the design flow. Since the method takes advantage of the RTL characteristics, it is simpler and requires less computational resources. For large, complex circuits, the method may perform disjoint partitions that, with low exactitude cost, allows to approach the circuit analysis with less computational resource load.

With reference to the future work, it can be grouped in four set:

1. The future work concerning **the better usage of the RTL advantages**
2. Concerning a **better implementation** of the proposed method
3. Regarding the **extension** of the proposed method
4. Concerning the **application** of the proposed method **in other digital design fields**

Next, the information about these four groups of future work is extended:

1. With reference to the future work concerning **the better usage of the RTL advantages**, firstly, a study about the optimum BDD ordering of the operands of each RTL operator could be done. Since the information about operands and operators is available in the EHM, the BDDs would be built with an optimum ordering. This approach has been already carried

out for some operators, such as *bigger than*, *addition* and *subtraction*. Therefore, the approach would have to be extended to other operators, such as multiplication and division. The experimental results chapter (chapter E3) showed that, thanks to this ordering, significant BDD size reduction have been achieved at RTL.

Another interesting study would take advantage of the RTL information to analyze the circuit functionality, what could simplify further analysis. Examples of this kind of analyses are shown in annex chapters AIII.5.1 and AIII.7.1 (in Spanish), where different abstraction levels inside the RTL are shown for the circuits. The functional analysis could help obtaining a higher abstraction level, what could help simplifying the analysis. In the cited examples, the analysis has been performed manually, but the objective is being able to automate the process.

Another area where RTL could contribute is the consideration of multi-bit activities and probabilities. Frequently, the probabilities and activities of multi-bit signals are referred to the signal considered as a whole. Considering the probabilities and activities bit by bit implies a loss of information. For instance, the transition sequence of a 2-bit counter is $0\rightarrow1\rightarrow2\rightarrow3\rightarrow0$.... Then, it is more exact to consider this sequence with its 2-bits probabilities and activities than the individual bit sequence, where the bits activities are: $a_{c(1)}$=0,5 y $a_{c(0)}$=1. To perform this analysis, the activity definition would have to be transformed, because there would not be only four possible transitions related to a bit ($0\rightarrow0$, $0\rightarrow1$, $1\rightarrow0$, $1\rightarrow1$), but a lot of different transitions depending on the signal width ($0\rightarrow0$, $0\rightarrow1$, $0\rightarrow2$, $0\rightarrow3$, $1\rightarrow0$, $1\rightarrow1$, $1\rightarrow2$, ...). Later, the new transition probabilities should be translated to the bit-level activity.

2. The future work considering a **better implementation** of the proposed method is development task. Since it is a first proposal, the method implementation in a computer tool is preliminary. As it has been said, the accepted VHDL subset is reduced; thus, the first step would be to extend the subset to at least the synthesizable subset.

   It would also be convenient to extend the analysis to hierarchical designs. At the moment, structural designs are flattened prior to the analysis. Since the model would not be modified, it would be just a technical improvement.

   As a result of these implementation improvements, large circuits could be analyzed in an easy way. This would allow to extend the experimental results and to analyze the model behavior in more unfavorable conditions.

3. A field that requires further research is the work related to the **model extension**. In this area there are some important considerations:

   - Considering **memory elements**. In order to make the model applicable, it is necessary to include sequential circuits. At gate level, several contributions consider memory elements [46], [133], [120], [5]; therefore, a first approach could adapt any of these contributions to the proposed model. The next step would investigate the RTL characteristics that facilitate the sequential circuit analysis. For example, obtaining the probabilities and activities of a signal from the VHDL process that defines the finite state machine. To make the most of this study, this could be done in conjunction with the consideration of multibit signals.
   - Considering **spatial dependences at inputs**. Some contributions have studied the spatial correlations at inputs [80], [7], [5]. At RTL, the usage of multibit signals and considering the activity as a whole (not bit by bit), provides a more accurate way for considering the spatial dependences between the bits of a signal. The usage of multibit signals has already been pointed out as future work. On the other hand, in a first step, spatial dependences between different signals could be treated similarly to the previous contributions, but considering multibit signals. Afterwards, it would be convenient to investigate other solutions that take advantages of the RTL characteristics.
   - Considering **spurious transitions** (*glitches*). Spurious transitions may considerably increase the circuit activity. Such transitions depend on the gate delay and on the

circuit's gate arrangement. Therefore, that information is not known at RTL because it depends on the synthesis tool, the technology library and the synthesis options. Furthermore, glitches are not even easy to estimate at gate level [113]. Thus, RTL estimation could not intend to know the influence of these transitions on the activity. Then, the model would be a zero-delay model, which will give the circuit minimum activity. Nevertheless, despite of the RTL limitations, a rough estimation of the spurious transitions could be performed considering the signals' combinational depth and how balanced the combinational paths are.

4. The application of the proposed method in **other digital design areas** could be very promising. Some of the different fields that could be developed are:

   - **Power estimation**: This is the most evident application because dynamic power directly depends on switching activity. Tools like Xilinx's *XPower* [142] estimate the power consumption of a circuit implemented in a FPGA from the activity data provided by the user. Therefore, obtaining the activity data would be an immediate application of the method

     In addition, dynamic power could be calculated estimating the circuit's gate arrangement by means of the EHM. Using this information and the internal nodes' activity, a power function depending on the internal gate capacitances could be obtained. Such capacitances would depend on the technology library and, in a more sophisticated model, on the fanout and the characteristics of each gate.

     Last, due to recent the importance of static power, knowing the switching activity could help finding the tradeoffs between dynamic and static power. Static power could be estimated using the EHM because it provides a circuit structure that may help estimating the total number of gates and their size, which are fundamental static power parameters. On the other hand, due to the dependence of static power on the logic values at the gate inputs [39], [25], getting the signal probabilities necessary to calculate the activity, would also contribute to the estimation of static power.

   - **Area estimation**: The circuit analysis and the elaboration of the EHM constitute an intermediate step towards the synthesis. Therefore, the EHM information joined with the BDD analysis could be used to estimate area. Favorable results have already been obtained using the SHM [76] (§E2.3.1); therefore, it lends support to the assumption that the model extension (EHM) will provide better results.

   - **Testability analysis**: Signal activity may be an indication of its controllability. Detecting a signal with low activity may reveal its low controllability, what is considered an important parameter in testability analysis. Besides, the EHM analysis could contribute to detect areas with low observability.

     On the other hand, the activity and probability values obtained could help identifying inadequate test vectors and also finding appropriate test vectors.

   - **Reliability analysis**: High activity values may compromise the circuit reliability due to the high temperatures caused by dynamic power. Besides, high activities could generate noise that could affect mixed signal circuits (analog and digital circuits). Knowing both circuit activity and circuit structure (provided by EHM) could contribute to detect hot areas and areas where it would not be advisable to have digital and analog signals near.

   - **Functional analysis**: BDDs provide information about signal functionality, in fact, the logic function of a signal can be obtained from its probability BDD. Therefore, the BDD analysis of a circuit in conjunction with the knowledge of the circuit structure provided by the EHM could contribute to the analysis of the circuit functionality.

   - **Quality analysis**: The SHM (Torroja [130]) extension, which results in the EHM, in conjunction with the functional analysis provided by the BDDs, provides a greater amount of quality analysis possibilities. The implementation of many of these

analyses is immediate. In fact, they have arisen during the experimental phase of this Thesis, where many of the cases that are cited next have been detected. These studies would contribute to extend the Department's research related to the quality analysis [20], [130], [132]. Examples of these analysis are:

- Detection of conditions that never are reached
- Detection of impossible conditions (conditions that always are false) and tautologies (conditions that always are true)
- Detection of useless code, that is, code that can be simplified by a constant, a signal or an expression simpler than the original
- Detection of signals that, although they apparently affects to another signal, they really do not affect

- **Functional verification**: To a certain extent, a signal's probability value reveals its functionality. The probability equation of a signal is equivalent to its logic function but instead of assigning binary values ($x_i \in \{0, 1\}$) to the function variables x_i, probabilities are assigned ($x_i \in [0, 1]$). That is, the only difference is that the function domain is no longer discrete but continuous. As a result, the image also will be continuous, which is $f \in [0, 1]$, instead of ($f \in \{0,1\}$).

  Assigning probability values instead of logic values could be comparable to performing an infinite simulation in which the input vectors have the assigned probabilities. As a result, the statistics of the signal being at '1' would be obtained, which are equivalent to the resulting probabilities. Since it is a probabilistic analysis and not a statistical one, the simulation time is infinite. Thus, the result does not depend on the input vector specific values used in the simulation. The difference is similar to the difference between static methods and dynamic methods used for activity estimation (see chapter E1).

  If extreme probabilities (values close to 0 or 1) are not assigned to the inputs, it is improbable for two similar circuits but not identical to have the same probabilities in their internal signals and output ports.

  This fact has been proved during the experimental phase of this Thesis. As it has been said in section E3.1, the RTL circuits used in the experimental phase have been manually adapted for the tool that performs the probabilistic analysis[79]. Among other tasks, the adaptation consists in the circuit flattening and the substitution of multibit signals into bit signals. Unfortunately, manual manipulations are error prone. In this case, mistakes such as assigning different vector indexes or forgetting to update after cut and paste some code lines, were usual. In order to assure that the circuit adaptation were correctly done; both gate level and RTL circuits were simulated and the waveforms of output ports and common signals were compared.

  During the comparisons, it was observed that whenever the simulation showed that the adaptation was erroneous, the probability values of the gate level and RTL analyses were different. That is, when a signal's gate level waveform was different from the RTL waveform, their probabilities resulting from both analyses were also different. This fact was corroborated in all the circuits: no case was found with different simulation values but with identical probabilistic values. The contrary was not also found; that is: no case was found with identical simulation values but with different probabilistic values. Certainly, since this hypothesis has only proved with few circuits, it requires a deeper analysis. Nevertheless, the results lend support the assumption that **the method could be applied to functional verification**.

[79] The gate level circuits were not manually adapted because the synthesis process can produce a flattened and bit level circuit

The method could be extended. For example, in order to confirm that two circuits are equivalent, the method could check with different probability values at inputs (what it would be comparable to different simulations). In addition, in case that no disjoint partition were made, activity values could also be compared because they also would be exact. This comparison would provide a higher confidence to the verification process; although, it does not seem that such a confidence is needed. However, for an even higher confidence, all the probability and activity values resulting from the mesh of figure E3-10 could be obtained. As a result, a kind of circuit signature with all its probability and activity values would be obtained.

The benefits of the probabilistic technique are evident, since it is much faster and convenient to perform an automatic probabilistic analysis rather than performing a simulation, in which the test-bench has to be created and the simulation times are larger. Furthermore, the probabilistic results are easier to check because it is a comparison between numbers and not waveforms. Finally, the EHM could assist in finding where in the circuit the differences are. Since EHM provides signal dependence trees and backannotation to the source code, the EHM could indicate the signals in which the differences are originated and how they are propagated.