Reliability Aware Logic Synthesis through Rewriting

Satish Grandhi†, Christian Spagnol†, Jiaoyan Chen†, Emanuel Popovici †, Sorin Cotafona*
† Department of Electrical and Electronic Engineering, University College Cork, Cork, Ireland
* Faculty of EE, Mathematics and CS, Delft University of Technology, Delft, The Netherlands
sagrand@ue.ucc.ie, christian.spagnol@ue.ucc.ie, chenj@ue.ucc.ie, e.popovici@ucc.ie, s.d.cotofana@tudelft.nl

***Abstract*—The low reliability of advanced CMOS devices has become a critical issue that has to be considered in the digital IC design flow. This paper introduces a design time methodology to address and improve the reliability of combinational circuits. The key idea is to employ local transformation rules, a methodology that were extensively used for area, delay, and power optimizations and demonstrate that they can reduce the error probability as well. We propose a set of local transformation rules that enhance the reliability without altering the circuit functionality. This functional rewriting capability, along with a circuit reliability assessment methodology developed in house, enables the integration of the reliability aware analysis and logic optimization algorithm that iteratively transforms the design in order to achieve higher circuit reliability. Experimental results based on simulations performed on MCNC benchmark circuits indicate that method can provide a reliability improvement of up to 7.5%.**

***Index Terms*—And-Invert Graphs (AIG), ABC Tool, Local Transformation Rules, Optimization, Reliability, Synthesis, Rewriting**

I. Introduction

Traditional logic synthesis methodologies and tools are centered on fulfilling timing, power, and area constraints or on achieving acceptable tradeoffs among those [1], [2]. However, as the CMOS technology entered the nanometer era by means of shrinking those metrics cannot cover any longer all the relevant aspects. Nanotechnology specific issues, e.g., v_{DD} reduction, higher impact of process parameter and temperature variations, result of increased device failure rates, making CMOS ICs less reliable [3] , [4]. As a result, reliability is turning out to be a major design metric sharing equal importance with the traditional ones. This paper investigates gate level methodologies to improve circuit reliability by employing logic synthesis techniques. Reliability driven logic synthesis is one area that is gaining lot of importance in the last few years. In [5], Soft Error Reliability (SER) is improved through localized circuit restructuring taking advantage of don't care based re-synthesis and local rewriting. In [6] , a technique to improve the circuit robustness to soft errors based on redundancy addition and removal (RAR) by eliminating gates with large contribution to the overall SER is proposed. Efficient algorithms for synthesizing approximate circuits for concurrent error masking of logical and timing errors was employed in [7]. ATPG-based rewiring method to generate functionally-equivalent yet structurally-different implementations to reduce the SER were developed in [8].

All these approaches employ redundant node addition techniques to improve circuit reliability. Our approach slightly differs as we use subset of NPN-equivalent (Negation-Permutation-Negation equivalent) logic configurations to improve circuit reliability. The biggest advantage is that we do not add any extra overhead by increasing node count. Though reliability driven logic optimization is in its infancy when compared to power and delay driven optimization, the method presented here is still based on the popular and successful concept of local transformations [9]. We introduce set of local transformation rules for logic optimization from a reliability perspective. We then introduce an algorithm to compute the impact of gate errors on the circuit output(s). Due to space limitations, only the final formulas are provided with no details of the mathematical analysis. We then develop reliability aware logic synthesis tool which applies the transformation rules in a guided fashion on complex combinational circuits. We have evaluated our tool on a set of MCNC benchmark circuits and results show a reliability improvement upto 7.5%.

This paper is organized as follows. Section 2 briefs about the reliability evaluation mechanism. Section 3 describes the the local transformation rules and its analysis. Section 4 presents the reliability aware logic synthesis flow. Section 5 discusses a case study and the simulations results based on MCNC benchmark circuits. Section 6 concludes the paper and provides directions for future work.

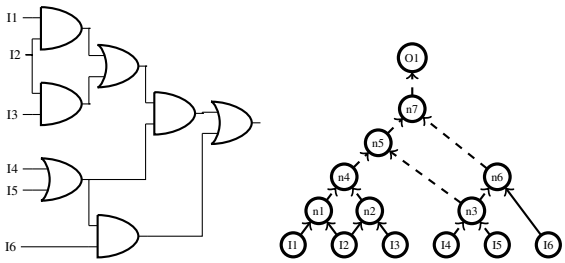
II. Reliability Evaluator

To built circuits optimized for reliability, we need a reliability evaluation engine to confirm that the resultant circuit is better than the original one. In this section, we first describe AND Inverter Graph (AIG) and then describe the reliability evaluation methodology.

A. AND Inverter Graph (AIG)

One of the major decisions in designing an EDA tool is the selection of the right data structure as it determines the speed and efficiency of the tool. Two of the most commonly used data structures in EDA for digital circuit synthesis are And Inverter Graph (AIG) and Binary Decision Diagrams (BDDs) [2]. AIG is a Boolean network composed of two-input-ANDs and inverters. Fig. 1 depicts a simple combinational circuit and its corresponding AIG representation. The circle represent the 2-Input AND gates and edges with a dash line indicate

negation i.e. inversion of that input. AIGs are preferred over BDDs to represent the circuits for various advantages it has got to offer [10], [11]. AIG unifies equivalence checking, synthesis and technology mapping and offer better performance and correlation with final area and delay once the circuit has been mapped to a target technology [11].



(a) Reference Circuit *(b)* AIG representation

Fig. 1. Reference Circuit and its AIG representation.

B. Reliability Evaluator

Reliability analysis of logic circuits deals with the computation of the impact that the gate errors implicate on the Primary Outputs (PO's) of the circuit. As we represent circuits in the AIG format, a novel algorithm based on probability principles is developed, with the prime focus being AND and INVERT gates. Due to space constraints, only the resultant equations are presented here without complete mathematical analysis. For complete details, refer to [12].

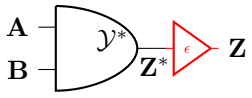


Fig. 2. Unreliable AND Gate Model

An unreliable AND gate can be modeled as an ideal (error free) AND gate followed by a faulty buffer that represents the stochastic behavior of the errors. The two nodes Z* and Z are named as internal and the external output node. The output node can be in error due to two possible reasons: (i) propagation of the errors onto the gate input nodes and (ii) intrinsic errors within the gate. If no reconvergence fan out is present, the error on the internal output node can be represented as in (1).

$$\begin{aligned} P_\epsilon(\mathbf{Z}^*) =& P_\epsilon(\mathbf{A})P_\epsilon(\mathbf{B})P_{\mathbf{A}}(0)P_{\mathbf{B}}(0)+ \\ & P_\epsilon(\mathbf{A})(1-P_\epsilon(\mathbf{B}))P_{\mathbf{A}}(0)P_{\mathbf{B}}(1)+ \\ & (1-P_\epsilon(\mathbf{A}))P_\epsilon(\mathbf{B})P_{\mathbf{A}}(1)P_{\mathbf{B}}(0)+ \\ & [P_\epsilon(\mathbf{A})+P_\epsilon(\mathbf{B})-P_\epsilon(\mathbf{A})P_\epsilon(\mathbf{B})]P_{\mathbf{A}}(1)P_{\mathbf{B}}(1) \end{aligned} \quad (1)$$

The error and the static probabilites on the external output node can be represented as in (2).

$$\begin{aligned} P_z(1) &= P_z^*(1)(1-P_F)+P_Z^*(0)P_F \\ P_z(0) &= P_z^*(0)(1-P_F)+P_Z^*(1)P_F \\ P_\epsilon(Z) &= P_F+P_\epsilon(Z^*)-2*P_F*P_\epsilon(Z^*) \end{aligned} \quad (2)$$

Alg. 1 presents the methodology employed within the tool to compute the circuit reliability. Using (1 and 2), the error due to the AND gate is computed both on the internal and external output nodes. This flow has been integrated into the open source tool 'abc' [13] and automates the error probability computation.

Algorithm 1 Generic Method for Reliability Evaluation

INPUT:N, total number of nodes in the AIG network, Error Probability of Individual Gates and Switching activity P_{SA} on the primary input nodes (PI's)
OUTPUT: Output error probability
  for all nodes I= 1 to N **do**
    if Input Nodes are inverted **then**
      Account for the inverter error
    end if
    Compute Internal node error probability using (1)
    Compute Output node error probability using (2)
  end for

III. Reliability Aware Rewriting

Rewriting is a common approach to perform logic optimization based on local transformations. We introduce set of AIG based local transformation rules that can be applied on any combinational circuit. We then study the impact of the gate error probability on equivalent logic configurations to determine the best realization.

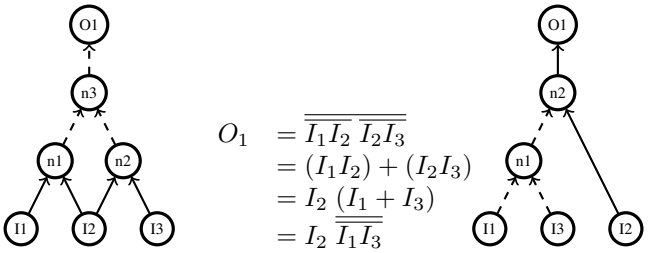
A. Local Transformation Rules

We now present a set of AIG based local transformation rules in our quest for reliability optimized implementation of combinational logic. The method presented in this paper is based on the successful technique of local transformations [14]. The five rules presented here are based on exhaustive matlab simulations performed to quantify the improvement in reliability. Though these rules are generic, an intelligent heuristic algorithm is employed with the constraint of improving the circuit reliability and not area/delay, the novelty lies in its application. These rules are not part of existing 'abc' tool and instead supplement the existing set of rules already available.

–Rule1: The first transformation as shown in Fig. 3 is modelled based on the law of distributivity. Consider reconvergent fanout node n3 with two fan-in nodes n1 and n2. If both n1 and n2 have no other fanouts except n3 then I1 and I3 can be swapped with necessary negations. The new node shall have higher reliability as well as node count decreases by one.

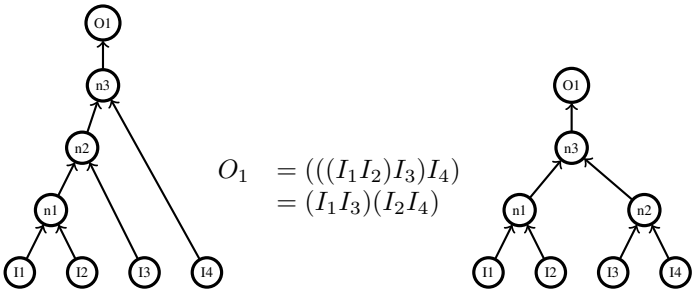
–Rule2: This transformation as shown in Fig. 4 is based on the law of associativity. The underlying assumption is that reducing the length of the longest path will improve the reliability of the circuit.

–Rule3: This rule as shown in Fig. 5 is also based on the law of associativity. It suggests equally distributed inverters on the both the legs of the graph will improve reliability of the circuit. This rule specifically targets the configuration for a 2



$$\begin{aligned} O_1 &= \overline{\overline{I_1 I_2}\ \overline{I_2 I_3}} \\ &= (I_1 I_2) + (I_2 I_3) \\ &= I_2\ (I_1 + I_3) \\ &= I_2\ \overline{\overline{I_1 I_3}} \end{aligned}$$

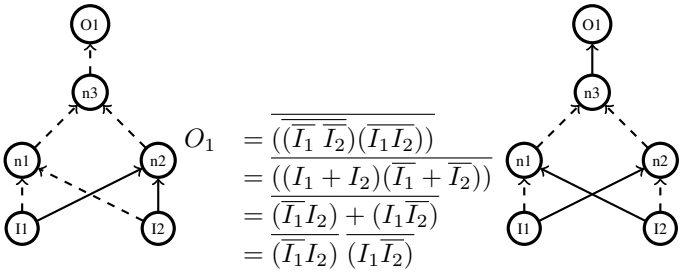
Fig. 3. Logic Transformation Rule1.



$$\begin{aligned} O_1 &= (((I_1 I_2) I_3) I_4) \\ &= (I_1 I_3)(I_2 I_4) \end{aligned}$$

Fig. 4. Logic Transformation Rule2.

input XOR gate. Its application can be seen predominantly in circuits like priority encoders, CORDIC processors, etc.



$$\begin{aligned} O_1 &= \overline{\overline{(\overline{\overline{I_1}\ \overline{I_2}})(\overline{I_1 I_2})}} \\ &= \overline{((I_1 + I_2)(\overline{I_1} + \overline{I_2}))} \\ &= \overline{(\overline{I_1} I_2) + (I_1 \overline{I_2})} \\ &= \overline{(\overline{I_1} I_2)}\ \overline{(I_1 \overline{I_2})} \end{aligned}$$

Fig. 5. Logic Transformation Rule3.

–Rule4: This rule as shown in Fig. 6 is derived based on principles of associativity and insertion and defines the best representation for 3 bit majority voter. In principle it first applies Rule1 to reduce node count and then applies Rule3 to distribute the inverters equally on both the legs of the output.

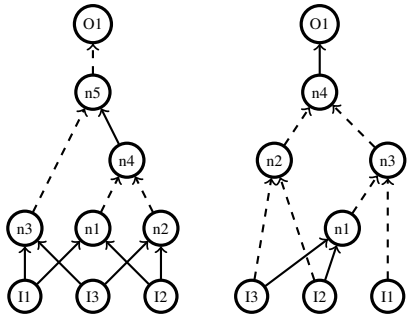
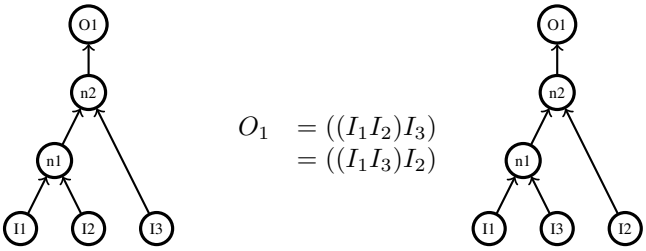


Fig. 6. Logic Transformation Rule4.

$$\begin{aligned} O_1 &= \overline{\overline{(I_1 I_2)}\ \overline{(I_2 I_3)}\ \overline{(I_3 I_1)}} \\ &= I_1 I_2 + I_2 I_3 + I_3 I_1 \\ &= I_3 I_1 + I_1 I_2 + I_2 I_3 + I_2 I_3 \\ &= I_3 I_1 + I_2 I_3 I_3 + I_1 I_2 + I_2 I_2 I_3 \\ &= I_3(I_1 + I_2 I_3) + I_2(I_1 + I_2 I_3) \\ &= (I_3 + I_2)(I_1 + I_2 I_3) \\ &= (\overline{\overline{I_3}\ \overline{I_2}})(\overline{\overline{I_1}(\overline{I_2 I_3})}) \end{aligned} \tag{3}$$

–Rule5: The fifth transformation rule as shown in Fig. 7 is based on the commutative law. The rule indicate that the signals with the lowest static probability of '1' in an AIG tree should be closer to the output, or closer to the root node of a sub graph. Intuitively, this rule is maximizing the masking effect of the AND gate to minimize the effect of any error coming from the other side of the graph. The reliability improvement is strongly dependent on the static probability of the input pins.



$$\begin{aligned} O_1 &= ((I_1 I_2) I_3) \\ &= ((I_1 I_3) I_2) \end{aligned}$$

Fig. 7. Logic Transformation Rule5.

B. Exhuastive Analysis of Rules

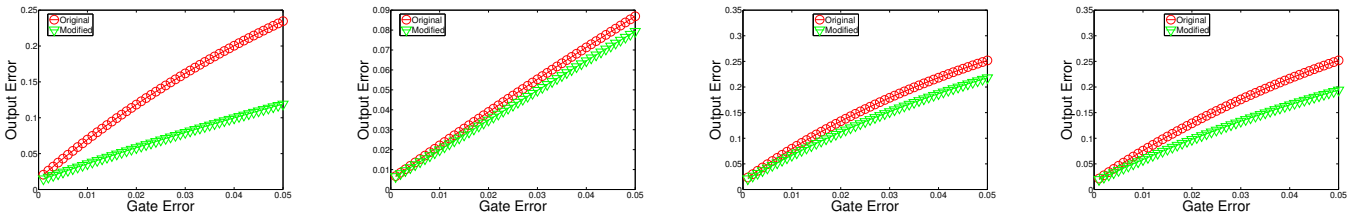
Each of the rewriting rules presented earlier is a transformation between two logical equivalent circuits. A mathematical equation that describes the primary output reliability as a function of the input error probabilities input static probabilities and the gate error probability can be associated with each of this logical equivalent circuit by recursively applying (1) and (2) for each node of the circuits. The problem then become chosing between:

$$\begin{aligned} R_{Org} &= f_1\{PE_I, SP_I, GE\} \\ R_{Mod} &= f_2\{PE_I, SP_I, GE\} \end{aligned} \tag{4}$$

where,

- R_{Org} - Reliability of the original circuit
- R_{Mod} - Reliability of the modified circuit
- PE_I - Error Probability of the Primary Inputs
- SP_I - Static Probability of the Primary Inputs
- GE - Individual gate error

Consider a three input graph. (4) have seven variables: three input error probabilities, three input static probabilities and the gate error probability. As an example of the complexity, apply (4) to Rule4. We obtain an equation of the output error probability that is a polynomial in seven variable of degree 20. It is hence clear that direct mathematical analysis is not feasible to formally compare the reliability performance

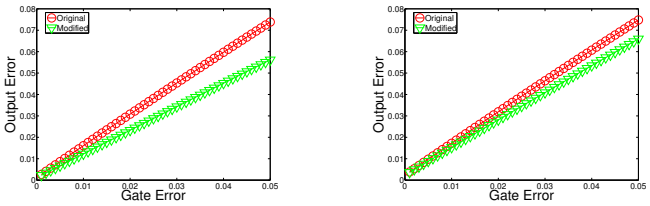


(a) Rule1 *(b)* Rule2 *(c)* Rule3 *(d)* Rule4

Fig. 8. Simulation results for Rule1-4.

of the circuit configurations from these equations. As an approximate solution, a framework has been developed in matlab to investigate the behaviour of the equivalent circuits. We apply the most commonly used signal patterns Gaussian, Inverse Gaussian, monotonically increasing and monotonically decreasing distributions across the input nodes for error probability and static probability and then evaluate (4) varying the gate error probability. Exhaustive analysis of these rules have been performed for different input static probability values and gate errors. Due to space limitation, only the results of one case Gaussian distribution of input error probabilities and constant values for static probability are presented here. But the discussion and conclusions are drawn from the exhaustive simulations done for many different patterns.

The simulation results are plotted in Fig. 8-9. Simulation results for different input patterns confirm that **rules [1-4]** improve the reliability of the circuit and hence suggest that they are applicable in any general scenario. In contrast, **rule5** is applicable only under certain circumstances based on the input static probability values. From the **Rule1** plots, we see that considerable reliability improvement is achieved. While performing various set of simulations, we observed that intelligent insertion of extra nodes can result in reliability improvement. The question of how to insert such extra nodes in a systematic fashion is the subject of future investigation.



(a) Pe_{IX}=0.01,SP_{I1}=0.2, SP_{I2}=0.1,SP_{I3}=0.5 *(b)* Pe_{IX}=0.01,SP_{I1}=0.15, SP_{I2}=0.3,SP_{I3}=0.5

Fig. 9. Rule5 Simulation results

IV. Reliability Aware Optimization

To validate the proposed approach and rules, a local optimization search algorithm is presented to quantify the reliability improvement. The aim is to find a sequence of transformations leading to an AIG network that minimise the cost function. We rely on a heuristic approach to find an acceptable solution, i.e., an AIG providing reliability higher than the original circuit. The strategy chosen is to apply local optimization based on the transformation ruleset. Given a combinational circuit implementing the Boolean function f and its AIG network, we traverse through the graph to see if any of the rules are applicable on the given node. For every possible transformation, the new reliability of the circuit is computed. The configuration that yields the highest improvement in circuit reliability is chosen and the new topology is generated. This process is continued on every node on the graph until we reach the primary outputs where no more transformations are applicable.

Fig. 10 describes the complete flow of the tool. The output error probability of two netlists is computed; (i)the default circuit (the original MCNC netlist) without any optimization and (ii)the circuit optimized by the best 'abc' synthesis algorithm. Since the synthesis algorithm on the abc tool focuses on area and delay optimization it may deteriorate the reliability of the circuit. After selecting the better circuit configuration between the two, the internal tool developed is employed to further improve the circuit reliability. The synthesis tool traverses through every single node in the circuit performing Boolean matching to see if there are any matching rewriting rules that can be applied. If there is more than one applicable rule, the one which provides the highest reliability improvement is selected. This process continues until no more rules can be applied on this node that can improve the circuit reliability. We perform similar set of operations on all the nodes in the circuit. This methodology will continue to benefit with further expansion of the local transformation ruleset.

V. Experimental Results

As a case study, the proposed reliability aware synthesis algorithm is applied on the AIG depicted in Fig. 1. As the circuit is simple, only two local transformations are applicable. **Rule2** is applicable on node n1. The new error probability of the circuit after applying this rule is presented in Fig. 12. It is clear that applying **Rule2** on node n1 results in higher reliability. No other rule is applicable on node n1. This is the new reference topology of the circuit. Also, **Rule2** can be applied on node n2 of Fig. 11(a). After this transformation, the reliability of the circuit reduces and hence this transformation is not applicable. Further, **Rule2** can also be applied on node n3 of Fig. 11(c). Simulation results presented in Fig. 12 show that such transformation also results in improvement in the

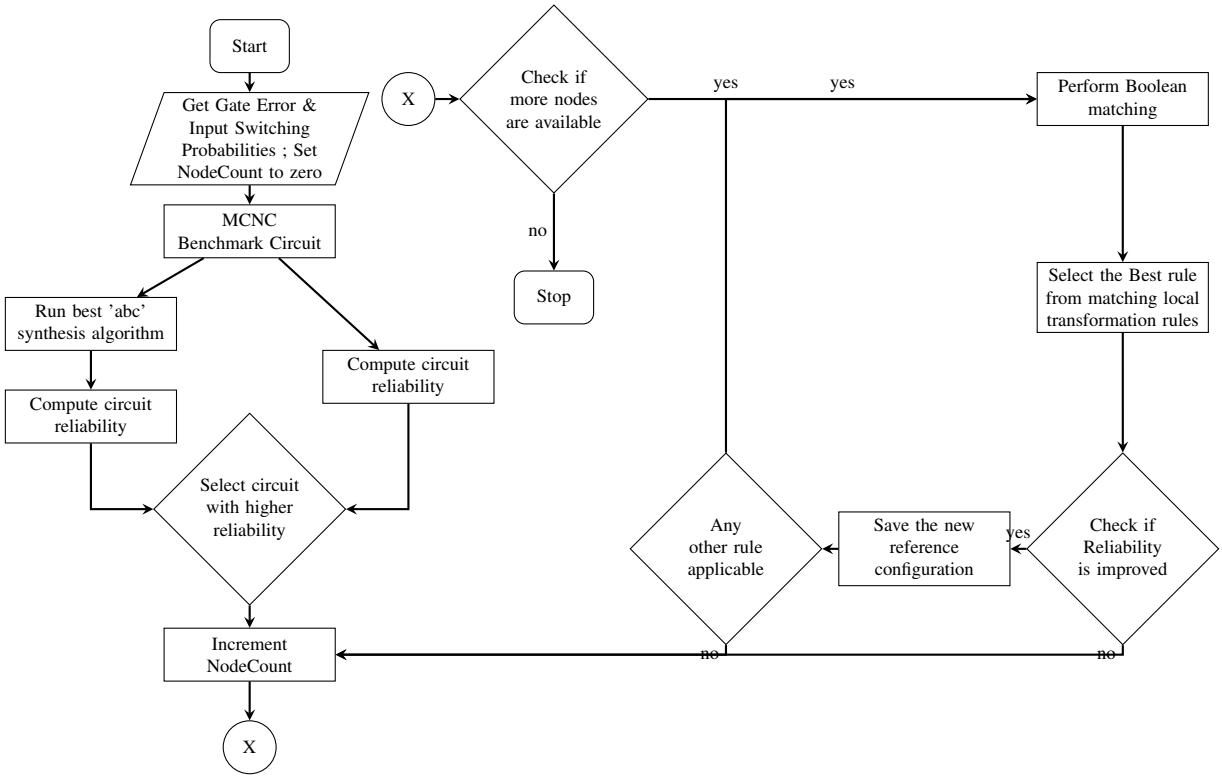


Fig. 10. Reliability Aware Logic Synthesis Flow

reliability of the circuit. No further rules can be applied on any of the nodes and this remain as the most optimized version of the reference circuit. Fig. 12 shows the improvement in the reliability numbers.

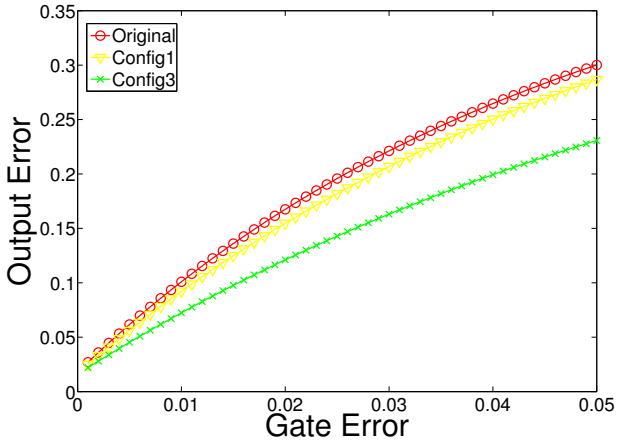


Fig. 12. Comparsion of results.

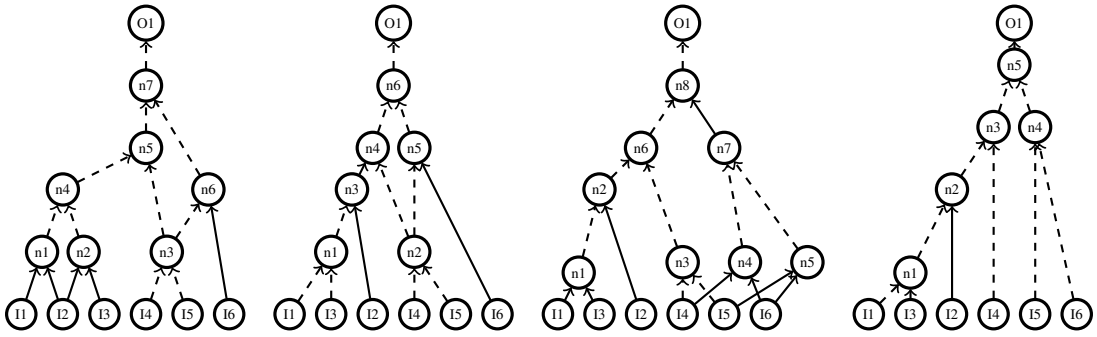
To prove that the whole methodology is scalable, we applied the tool on a set of MCNC benchmark circuits. Simulation results comparing the circuit reliability of default and the optimized configuration obtained from our tool are reported in Tab. I. In the table, columns 1 and 2 give the name and number of gates in the benchmark circuit. The reliability of both the original configuration and the optimized one are computed and tabulated in third column. The fourth column reports the percentage improvement achieved through our synthesis algorithm. Column 5 lists the total number of output nodes and those reliability improvements is greater than 0.5%. The reliability improvement is computed using (5).

$$R_{Metric} = \sum_{i=1}^{n} \frac{R_{Org} - R_{Mod}}{R_{Org}} \tag{5}$$

where 'n' is the number of nodes. The maximum improvement in reliability is 7.5% for the benchmark circuit, *x2*. Complex circuits with higher output node count include many paths with gate count less than 10 and generally there are no local transformation rules that are found suitable for application. Hence, *vda* and *pair* circuits with higher gate count report lower improvement in reliability.

VI. Conclusions and Future Work

This paper describes a technique to study the impact of gate failure on complex combinational circuits. We proposed a set of AIG based local transformation rules that improves circuit reliability. A synthesis algorithm was also presented that optimizes the circuit reliability. The main focus of the paper was to improve circuit reliability by employing traditional rewriting techniques. Applying the synthesis algorithm on the MCNC benchmark circuits with node count from 30 to 1500 resulted in improving overall circuit reliability by up to 7.5%. Due to



(a) Original *(b)* First transformation *(c)* Second transformation *(d)* Final

Fig. 11. Application of logic transformation ruleset on the AIG Representation shown in Fig. 1

Benchmark	GateCount	Output Error Probability		Reliability Improvement $R_{Metric}\%$	Output Node Details	
		Original	Optimized		Total	$R_{Metric} \geq 0.5\%$
b9	99	0.16023	0.15036	6.15808	20	7
cm162a	33	0.22993	0.21427	6.81026	5	4
cm85a	35	0.20816	0.19640	5.65277	3	2
cu	45	0.13332	0.12700	4.73912	5	5
dalu	1371	0.32429	0.31516	2.81394	16	15
frg1	125	0.17372	0.17089	1.62925	3	1
pair	1500	0.20542	0.20429	0.54835	131	28
unreg	112	0.09779	0.09365	4.23406	16	16
vda	924	0.15885	0.15724	1.01155	39	18
x2	60	0.16726	0.15468	7.51923	7	6

TABLE I
MCNC BENCHMARK CIRCUITS PERFORMANCE EVALUATION FOR DIFFERENT GATE ERROR (ϵ = 0.05.)

small number of local transformation rules, the improvement is sometimes marginal.

Going forward, we plan to extend the local transformation rule set to encompass more possible scenarios. From a reliability perspective, we believe that the AIG data structure is more appropriate to represent combinational circuits. In particular the fact that AIG are non canonical (i.e. there exist more graphs representing the same logic function) can be exploited to further improve reliability. We intend to extend the reliability evaluator to any generic circuit that would enable to report all the generic data like area, delay and power as well. We intend to test our tool on the more recent and complex IWLS 2005 benchmarks.

VII. ACKNOWLEDGEMENT

This work was supported by the Seventh Framework Programme of the European Union, under Grant Agreement number 309129 (i-RISC project).

REFERENCES

[1] S. Iman and M. Pedram. Pose: power optimization and synthesis environment. In *Design Automation Conference Proceedings 1996, 33rd*, pages 21–26, 1996.

[2] Rashmi Mehrotra, Tom English, Michel Schellekens, Steve Hollands, and Emanuel Popovici. Timing-driven power optimisation and power-driven timing optimisation of combinational circuits. *Journal of Low Power Electronics*, 7(3):364–380, 2011.

[3] S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *Micro, IEEE*, 25(6):10–16, 2005.

[4] C. Constantinescu. Trends and challenges in vlsi circuit reliability. *Micro, IEEE*, 23(4):14–19, 2003.

[5] Smita Krishnaswamy, Stephen M Plaza, Igor L Markov, and John P Hayes. Enhancing design robustness with reliability-aware resynthesis and logic simulation. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pages 149–154. IEEE, 2007.

[6] Kai-Chiang Wu and Diana Marculescu. A low-cost, systematic methodology for soft error robustness of logic circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(2):367–379, 2013.

[7] Mihir R Choudhury and Kartik Mohanram. Low cost concurrent error masking using approximate logic circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(8):1163–1176, 2013.

[8] Sobeeh Almukhaizim, Yiorgos Makris, Yu-Shen Yang, and Andreas Veneris. Seamless integration of ser in rewiring-based design space exploration. In *Test Conference, 2006. ITC'06. IEEE International*, pages 1–9. IEEE, 2006.

[9] J.A. Darringer, William H. Joyner, C.Leonard Berman, and Louise Trevillyan. Logic synthesis through local transformations. *IBM Journal of Research and Development*, 25(4):272–280, 1981.

[10] Robert Brayton and Alan Mishchenko. Abc: An academic industrial-strength verification tool. In *Computer Aided Verification*, pages 24–40. Springer, 2010.

[11] Thiago Figueiro, Renato P Ribas, and André Inácio Reis. Constructive aig optimization considering input weights. In *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pages 1–8. IEEE, 2011.

[12] Satish Grandhi, Christian Spagnol, and Emanuel Popovici. Reliability analysis of logic circuits using probabilistic techniques. In *Research in Microelectronics and Electronics, 2014. PRIME 2014. Ph.D.* IEEE, 2014.

[13] A Mishchenko et al. Abc: A system for sequential synthesis and verification. *URL http://www. eecs. berkeley. edu/˜ alanmi/abc*, 2007.

[14] A. Mishchenko, S. Chatterjee, and R. Brayton. Dag-aware aig rewriting: a fresh look at combinational logic synthesis. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 532–535, 2006.