GROK-LAB: Generating Real On-chip Knowledge for Intra-cluster Delays Using Timing Extraction

Benjamin Gojman
Department of Computer and Information Systems
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19104
bgojman@seas.upenn.edu

Sirisha Nalmela
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
snalmela@juniper.net

Nikil Mehta
Department of Computer Science California Institute of Technology MC 305-16
1200 E. California Blvd.
Pasadena, CA 91125
nikil@caltech.edu

Nicholas Howarth
nhowarth@seas.upenn.edu

André DeHon
andre@acm.org

Department of Electrical and Systems Engineering
University of Pennsylvania
200 S. 33rd St. Philadelphia, PA 19104

ABSTRACT

Timing Extraction identifies the delay of fine-grained components within an FPGA. From these computed delays, the delay of any path can be calculated. Moreover, a comparison of the fine-grained delays allows a detailed understanding of the amount and type of process variation that exists in the FPGA. To obtain these delays, Timing Extraction measures, using only resources already available in the FPGA, the delay of a small subset of the total paths in the FPGA. We apply Timing Extraction to the Logic Array Block (LAB) on an Altera Cyclone III FPGA to obtain a view of the delay down to near individual LUT granularity, characterizing components with delays on the order of a few hundred picoseconds with a resolution of ±3.2 ps. This information reveals that the 65 nm process used has, on average, random variation of $\sigma/\mu = 4.0\%$ with components having an average maximum spread of 83 ps. Timing Extraction also shows that as V_{DD} decreases from 1.2 V to 0.9 V in a Cyclone IV 60 nm FPGA, paths slow down and variation increases from $\sigma/\mu = 4.3\%$ to $\sigma/\mu = 5.8\%$, a clear indication that lowering V_{DD} magnifies the impact of random variation.

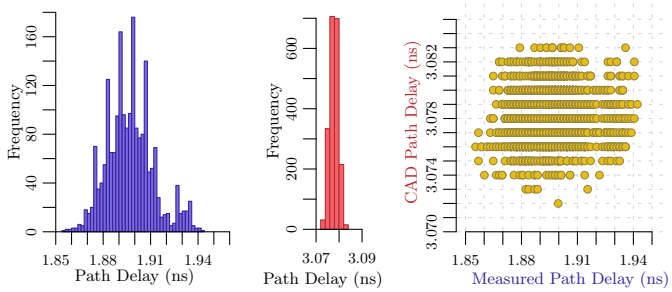
Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*placement and routing*; B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Algorithms, Measurement, Reliability

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
FPGA'13, February 11–13, 2013, Monterey, California, USA.
Copyright 2013 ACM 978-1-4503-1887-7/13/02 ...$15.00.



(a) Measured (b) CAD (c) Correlation

Figure 1: Path delay of 1000 nearly identical paths of length 7 LUTs, comparing measured delays to delays reported by the CAD tools for a Cyclone III 65 nm FPGA

Keywords

Component-Specific Mapping; Variation Measurment; Variation Characterization; In-System Measurement

1. INTRODUCTION

Circuit variation is quickly becoming one of the biggest problems to overcome if the benefit from Moore's Law scaling is to continue. It is no longer possible to maintain an abstraction of identical devices without incurring huge yield losses, performance penalties, and high energy costs. Current techniques such as margining and speed grade binning are used to deal with this problem. However, they will become prohibitively conservative, only offering a limited solution that will not scale as variation increases.

Fig. 1 concretely demonstrates the price we pay for these techniques. We carefully measured 1,000 paths consisting of seven buffers in one logic array block (LAB) of an Altera Cyclone III 65 nm FPGA. Fig. 1a shows a histogram of the results of these measurements. Similarly, Fig. 1b shows the distribution of delays as computed by the CAD tools for these paths. Observe that the mean of the measured distribution is significantly lower than that reported by the CAD tools. This illustrates the magnitude of conservative margining, showing that the fabricated paths are only 60%

the delay predicted by the CAD tools. Moreover, the measured distribution has a much larger spread — 96 ps vs. 11 ps. Fig. 1c demonstrates there is no correlation between the delays measured and those reported by the CAD tools.

FPGAs have the unique advantage over ASICs that they can use more fine-grained and aggressive techniques that carefully choose which resources to use after fabrication in order to mitigate adverse variation effects. In [10] we show that a component-specific mapping solution reduces energy needs by 50% and will be a necessity to extend beneficial scaling as variation increases. This approach requires measurement of the underlying resource delays for the CAD tools to generate a custom mapping perfectly adapted to the variation in the FPGA. In this paper we present Timing Extraction, a methodology that allows the kind of fine-grained measurement of fabricated component delays necessary for [10] in an efficient and inexpensive manner, utilizing only resources already available on conventional FPGAs. To practically validate Timing Extraction, we apply it to clusters (LABs) in the Altera Cyclone III and Cyclone IV FPGAs and confirm that the measurements and calculations reflect underlying process variation.

The key challenge in Timing Extraction is that it is not possible to directly measure the characteristics of every LUT or wire in an FPGA. Nonetheless, we show that it is possible to obtain fine-grain delays using an indirect approach to measure, compute and characterize the variation of small groups of components. Work in [18] demonstrated the feasibility of measuring path delays without the need of any dedicated test circuitry, by surrounding the path with two registers that are already part of the reconfigurable fabric. Timing Extraction takes advantage of this measurement technique but goes further by demonstrating how to use the measurements to resolve the delays of individual resources.

The measured path is composed of multiple components, the individual delays of which we would we would like to know. By configuring and measuring a small set of overlapping paths, we can setup a linear system of equations that, when solved, gives the individual delay of each component in the paths [5]. A simple example will give better intuition as to what the technique actually accomplishes. Consider that we measure three paths. Path 1 composed of component A and B. Path 2, B and C. Finally, Path 3, C and A. Suppose the delays of the paths are 5ps, 4ps and 3ps respectively. That leads to the system of equations below:

$$\begin{array}{ll} A + B = 5ps & \text{Path 1} \\ B + C = 4ps & \text{Path 2} \\ C + A = 3ps & \text{Path 3} \end{array}$$

Even though we did not measure the delays directly, with little work we can solve for the delay of A, B, and C to be 2ps, 3ps and 1ps respectively.

Timing Extraction does exactly this but at a level that allows us to characterize a full FPGA. Formulating the naive problem, where every wire and transistor in the FPGA is represented by a separate variable in the system of equations, invariably leads to an underdetermined system without a unique solution (Sec. 3.2). However, Timing Extraction judiciously groups components into discrete units of knowledge (DUKs) which, combined with a careful selection of measured paths, guarantee a solution to the delay of each DUK in the system (Sec. 3.3). With that information, we can predict the delay of any path that could be used when mapping logic to the FPGA.

We begin with a brief review of the required background (Sec. 2). Sec. 3 develops the ideas of Timing Extraction by using the Cyclone III as a case study. Results from our measurements are presented in Sec. 4. While we present concrete details on how to measure the Cyclone III, the general technique can be extended to any modern FPGA; in Sec. 5 we briefly sketch how to port the ideas and why they are generally applicable. An outline of future work is explored in Sec. 6, before concluding (Sec. 7).

Novel contributions of this work include:

- First identification and demonstration of techniques for determining the delay of individual LUTs and the unique interconnect delay between pairs of LUTs using only on-chip FPGA resources.
  - Identification of smallest delay-measurable groups of components
  - Identification of smallest set of measurements necessary to extract complete fine-grain delay information within a cluster (LAB)
  - Algorithm for calculating component delays from path measurements
- Technique for predicting delay of any path in a cluster (LAB) using component LUT delay measurements.
- First set of measurements to fully characterize the delay components within a cluster (LAB) in a commercial FPGA.
- Quantification of process variation at a near LUT-level granularity.
- Quantification of increased random variation with voltage scaling.
- Characterization of significant contribution from random variation in process variation.

2. BACKGROUND

2.1 Process Variation

Process variation refers to differences between device parameters due to manufacturing. These differences ultimately affect the delay and energy requirements of the device. Correlated variation has historically comprised the majority of process variation, where the amount a device varies is correlated to some parameter, such as location on the wafer. Consequently, most techniques aim to reduce correlated variation. Binning, for example, mitigates die-to-die variation, while biasing reduces correlated regional variation. In essence, correlated variation provides a model which can be used to reduce process variation. However, as feature sizes continue to shrink, more and smaller transistors fit on one chip, greatly increasing the contribution of random variation to process variation. Unfortunately, unlike correlated, random variation is not easily modeled and mitigated.

Fig. 2 shows how the three main contributors to random variation – oxide thickness, line edge roughness, and random dopant fluctuations – lead to a significant increase in variation experienced by V_{th}, the transistor's threshold voltage, as technology scales.

The value of V_{th} has a direct and profound effect on the performance and energy requirements of a transistor. Eqs. 1 and 2 represent the current through a transistor during the saturation and subthreshold operating points [6, 11]. Although physical parameters such as transistor geometry, W, L, and dopant concentration, η, have a strong stochastic

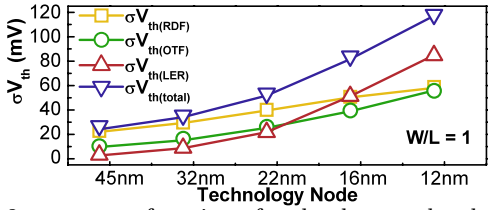


Figure 2: $\sigma_{V_{th}}$ as a function of technology nodes, based on predictive technology models. Considering the individual effects of random dopant fluctuations (RDF), line edge roughness (LER) and oxide thickness (OTF) from [19]

variation component, it is the exponential dependence on V_{th} that brings about the harmful effects of random variation on the current through a transistor.

$$I_{ds,sat} = W v_{sat} C_{ox} \left(V_{gs} - V_{th} - \frac{V_{d,sat}}{2} \right) \quad (1)$$

$$I_{ds,sub} = \frac{W}{L} \eta C_{ox} (n-1) \cdot {v_T}^2 \cdot e^{\frac{V_{gs}-V_{th}}{n \cdot v_T}} \left(1 - e^{\frac{-V_{ds}}{v_T}} \right) \quad (2)$$

In turn, the propagation delay τ_{pd} and leakage energy of the circuit are a function of current (Eqs. 3, 4).

$$\tau_{pd} = C_l \cdot \frac{V_{ds}}{I_{ds}} \quad (3) \qquad E_{leak} = I_{ds,sub} \cdot V_{ds} \cdot \tau_{cycle} \quad (4)$$

As such, random physical variation expresses itself in differences in the energy efficiency and delay of a transistor.

Statistical static timing analysis (SSTA) [14] attempts to model the expected random variation and with it the expected behavior of the FPGA. With this model, the CAD tools can generate a mapping that, statistically speaking, will reduce the effects of random variation. Unfortunately, this solution inherently fails to accommodate every FPGA. Instead of employing this one-size-fits-all solution, Timing Extraction measures and extracts detailed delay information from the FPGA after fabrication. This can then be provided to the CAD flow which generates a component-specific mapping tailoring the design to the particular FPGA.

The delay of a component in the FPGA is not only affected by process variation but can also fluctuate due to environmental and temperature changes [7] as well as aging effects [15]. To ensure that measured delays consistently represent process variation, Timing Extraction requires that measurements be taken in a highly controlled manner. Sec. 4.1 details the controls employed for our application on the Cyclone FPGA. The consistency of the results presented in Sec. 4.3 concretely demonstrates that Timing Extraction does measure process variation.

2.2 Altera Cyclone LAB Architecture

Timing Extraction is a general methodology that provides fine-grain delay measurement of small groups of components within an FPGA. Although it is applicable to any FPGA, to ground the presentation in this paper, we focus our application to the logic array blocks (LAB) of the Altera Cyclone III and Cyclone IV FPGAs.

The LAB in these FPGAs is composed of 16 Logic Elements (LE) each having a 4-LUT and optional register output, a set of 38 routing channels for external inputs, and 16 local routing channels for LE-to-LE communication with 50% depopulation (Fig. 4). The scope of this paper limits delay measurements to the 16 LEs and the 16 local routing channels in the LAB.

To better understand the results presented later in Sec. 4, it is worth noting that the architecture of the LUTs is such that nominally, the first two inputs, A and B, have similar delays and by design are slower than input C which in turn is tailored to be slower than input D. Moreover, inputs A and B form a complete input set, where every LE can connect to every other LE in the LAB by using either input A or B, and similarly inputs C and D form a complete input set.

2.3 Path-Delay Measurements

We use a launch-capture technique to measure the delay of a path in an FPGA. In this approach, a combinatorial circuit, known as the circuit under test (CUT), is configured between a launch register and a capture register. Starting at an initial frequency and increasing to a maximum frequency, signals are sent from the launch register to the capture register. When a signal fails to reach the capture register within half of a clock cycle, we know that the delay of the path is greater than twice the frequency at which that signal was clocked. This technique has been successfully used to capture the delay of paths on FPGAs for many applications [8, 12, 13, 18].

A limitation of this measurement technique, however, is that it cannot measure a path that is faster than twice the highest frequency supported by the FPGA's on-chip PLLs. Twice the frequency comes from the fact that the launch and capture registers are clocked on opposite clock edges. Therefore, any work that exclusively uses this measurement technique will be limited to reporting delays of long paths. To ground this, consider that the maximum frequency for the Cyclone III PLLs used in this work is 402.5 MHz. This means that the fastest path we can measure is $\frac{1}{2 \cdot 402.5} = 1.24$ ns. Fig. 1a shows that, on average, a path of length 7 LUTs is measured to take 1.90 ns, meaning that, roughly on average, the delay through one LUT is 271 ps. Combining this fact with our maximum frequency leads to the conclusion that the smallest path we can measure is 5 LUTs long. This ignores the expected variation spread. Therefore, to err on the side of caution, we do not measure anything with less than 6 LUTs in a path. Nevertheless, as we will later show, this work reports on delays on the order of one LUT by taking delay measurements of long paths and breaking them into smaller parts. [18] and [17] take only a single measure within each LAB or CLB and make no attempt to characterize within-LAB variation. The most closely related technique used in [3] and [20] takes the difference between two ring oscillators to extract subcluster delays. However, this approach fails to account for the unique interconnect delay between pairs of LUTs, nor is it able to account for register delays.

Due to the nature of CMOS and FPGA circuit design that uses NMOS pass transistors, there is a marked delay difference in a rising transition, as compared to a falling transition. In order to separate the falling and rising delays, our CUT is composed of buffers in series. In this way, all elements in a path transition in the same direction, allowing us to separate the rising transition through the path from falling transitions (Fig. 14). Fig. 3 shows a diagram of the path-delay measurement circuit used. A signal with a 50% duty cycle is provided to the launch register. The signal propagates through the CUT and the capture register records its output. Errors are detected by the two error detection circuits, one monitoring rising failures, the other, falling failures.

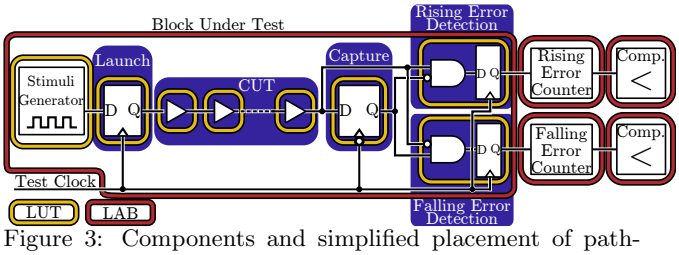


Figure 3: Components and simplified placement of path-delay measurement circuit

Because of operating variation such as clock jitter, it is not sufficient to observe one failure to declare the delay of a path. Instead, the path is tested at one frequency many times, and two counters, for rising and falling transitions, keep track of how many failures occurred at that frequency, for that transition. If at frequency f, the number of failures reaches a percent of the total number of transitions, the delay of that circuit is reported as $\frac{1}{f}$. The transition from no failures to 100% failures is gradual. If we assume that the variation that caused this gradual failure rate is mostly stochastic and has a symmetric probability distribution, then the 50% failure rate provides the most accurate estimate of delay given a small number of samples. We do not use this frequency for regular operation, since at this frequency signals fail timing 50% of the time. Knowing the variance in cycle time, we can then select a suitable operating frequency that keeps timing errors down to an acceptable level.

3. TIMING EXTRACTION

The general idea behind Timing Extraction is easy to understand. It is not possible to measure the delay of every component in an FPGA directly since individual transistors or wires cannot be isolated from their surrounding components. Nevertheless, by measuring the delay of different paths through an FPGA, it is possible to decompose the delays of these paths into their constituents. Essentially, each path consists of a linear sum of the delay of its parts; therefore, we can cast this problem as a linear system of equations where each equation represents a path and equals the measured delay of the path. With enough equations, we can solve for all the unknowns and directly acquire the delays of every component used in these paths. In order for the system of equations to have a unique solution, it is imperative to carefully select what the variables in the equations represent. In this section, we use the Altera Cyclone LAB architecture to ground the development of the general Timing Extraction methodology. We begin by considering what is individually calculable, followed by an analysis of what paths must be measured. This leads to the realization that our initial assessment of what is individually calculable is flawed, which ultimately arrives at the notion of discrete units of knowledge (DUKs), allowing for a complete solution.

3.1 Logical Components

It is not possible to measure the delay of a single wire or transistor in the FPGA, even indirectly. To explain, consider the simple representation of the Cyclone LUT in Fig. 4. Suppose we want to know the delay of only the highlighted crosspoint in isolation. This is not possible since any path that uses that crosspoint must use the labeled Local Interconnect, Output and MUX. However, since any path that uses this crosspoint will naturally use the other components,

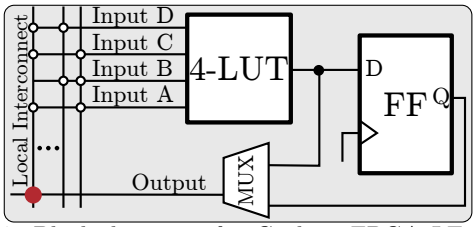


Figure 4: Block diagram of a Cyclone FPGA LE (4-LUT and register), including local interconnect

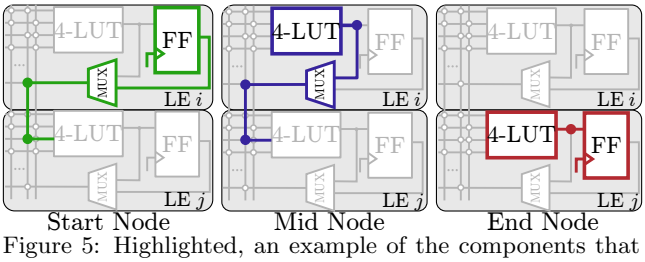


Figure 5: Highlighted, an example of the components that form each of the three types of LC Nodes in a Cyclone LAB

there is no practical reason to measure its delay independent of these components. This gives the notion of a Logical Component or LC Node, and the first attempt at defining what the variables in our system of equations represent.

As explained in Sec. 2.3, measured paths start at a register, go through zero or more buffers, and end at a register. A path in a LAB will begin at a register, go through some number of LUTs and end at a second register. Fig. 5 shows how we decompose this path into three types of LC Nodes. The path begins at an LC Node whose first component is a register, known as a *Start Node*, goes through zero or more LC Nodes with no registers, *Mid Nodes*, and ends at an *End Node*, an LC Node whose last component is a register.

Fig. 6a represents a path using groups of Start, Mid and End Nodes. Thus, we let LC Nodes correspond to variables in our system of equations and represent each measured path delay by a linear sum of the delays of these LC Nodes.

To solve for the delay of all LC Nodes, we must measure at least a number of paths equal to the number of LC Nodes in a LAB. A Start Node and Mid Node start at one LE and end at a second LE. Considering there are 16 LEs in a LAB and two input sets (Sec. 2.2), this gives a total of $16 \times 15 \times 2 = 480$ Start and 480 Mid Nodes per LAB. Since End Nodes only use one LE, there are only 16 End Nodes per LAB. In total, there are $480 + 480 + 16 = 976$ LC Nodes in a LAB, which is the minimum number of paths we must measure to solve for their delay.

3.2 Matrix Representation

Once we measure a correct set of 976 paths and solve for the delay of all LC Nodes, it will be possible to reconstruct the delay of any of the approximately 10^{18} paths within a LAB. Therefore, the problem is deciding which 976 paths to measure. To better discuss this solution, we formulate our system of equations as a matrix. A path is represented by a row, while a column describes an LC Node. An entry L_{ij} in the matrix is 1 if LC Node j forms part of path i, 0 otherwise. Since there are 976 LC Nodes, and we need at least 976 paths, our matrix will be at least as large as 976×976. Once the delays of the paths are measured, we use this matrix and the path delays to solve for all LC Nodes.

Linear algebra tells us that if the rank of the original matrix is equal to the number of LC Nodes, then we can solve

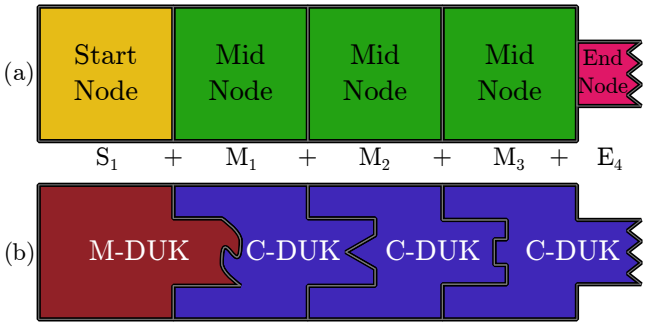


Figure 6: Equivalence between LC Node basis and DUK basis. To build intuition, the shapes give a geometric interpretation to the delay of each LC Node or DUK. The Equations below each figure show it mathematically

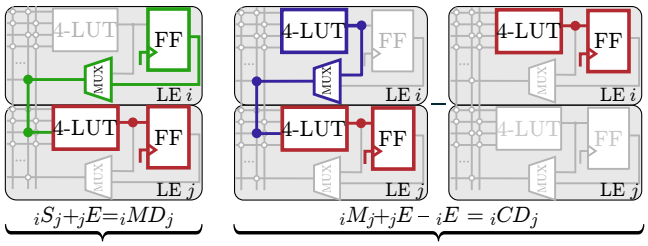
for the delay of each LC Node. Otherwise, if it is less than the number of LC Nodes, the system is underdetermined and, in general, contains an infinite number of solutions. Unfortunately, even if we measure the delay of all 10^{18} paths, the rank of the matrix is 960, 16 less than the total number of LC Nodes in a LAB. Sec. 5 provides some intuition as to why this is the case for any FPGA in which we let LC Nodes represent the variables in the system of equations.

Even though the matrix is rank deficient, it must have a non-empty vector space which comprises its basis. In turn, this means that there must be a set of linearly independent paths, which, when taken together and measured, allow us to compute the delay of any other measurable path in the circuit. Since the LAB has a matrix with rank 960, we only need to measure a linearly independent set of 960 paths to compute the delay of any path in the LAB. Essentially, instead of using a basis where every path in the matrix is represented by a linear combination of LC Nodes, we use a basis where every path is represented by a linear combination of the 960 paths measured.

Although this approach provides the delay of any path, it does not achieve the desired results for two reasons. First, it is difficult to incorporate these results into conventional routing algorithms when a component-specific route is sought, since routing algorithms [9] tend to expand routes incrementally and we only have complete path delay information. Second, the basis does not provide a fine-grained understanding of the variation. The next section addresses these shortcomings by defining a particularly convenient basis that spans the matrix yet provides the fine-grain, incremental variation information desired.

3.3 DUK Basis

Timing Extraction's objective is to provide fine-grain delay information that can then be used to characterize the variation in the FPGA as well as perform a component-specific mapping to the FPGA. We know it is not possible to solve for the delay of every LC Node; however, our solution should allow us to formulate path delays as a linear sum of a small number of components. By definition, an LC Node is the smallest delay we care to measure; however, since we cannot solve for LC Nodes, we consider the next best thing, a basis where the variables represent a small linear combination of LC Nodes. We refer to this small linear combination of LC Nodes as a *Discrete Unit of Knowledge*, or *DUK*. First we introduce the vectors that compose the



(a) M-DUK (b) C-DUK

Figure 7: Highlighted, an example of the LC Nodes that form the two types of DUKs in a Cyclone LAB

DUK basis, then we show the equivalence between an LC-based and a DUK-based model, finally we demonstrate that unlike LC Nodes, we can compute the delay of DUKs.

Instead of having three types of variables which are combined to represent a path, this basis contains two types of DUKs. The delay of a Start Node plus an End Node forms the first DUK (Eq. 5). On its own, this DUK forms a complete measurable path, starting at a register and ending at a second register. Moreover, all paths stem from this DUK, therefore, we refer to it as a Mother DUK, or *M-DUK*. The second DUK is known as a Child DUK, or *C-DUK*. As its name suggests, it follows the Mother DUK and incrementally grows a path. A C-DUK consists of the delay of a Mid Node plus the difference of two End Nodes (Eq. 6).

$$\text{M-DUK} = S_i + E_j \quad (5)$$

$$\text{C-DUK} = M_i + E_j - E_k \quad (6)$$

Assuming we have their delays, together, these two types of DUK allow us to compose any measurable path in exactly the same way that LC Nodes did. In general, a measurable path will be represented by an M-DUK and zero or more C-DUKs. For a path to be measurable, it must start and end at a register, M-DUKs naturally represent such paths. The function of a C-DUK is to replace the End Node and extend the path by adding a Mid Node and a new End Node. Consider, for example, the path shown in Fig. 6a consisting of a Start Node, 3 Mid Nodes and an End Node. We can easily represent this path in the DUK basis using one M-DUK and 3 C-DUKs, as shown in Fig. 6b. Fig. 6b represents each DUK as a jigsaw piece to give a geometric meaning to the notion that two DUKs must complement each other in order to correctly represent a path. Here, instead of each DUK having a different delay, each DUK has a unique shape. The concave left side of a C-DUK represents the carved out delay of the subtracted End Node, while the convex right side of a DUK shows the addition of an End Node.

In general, given a path represented by LC Nodes, we can easily re-express it using the DUK basis by replacing the Start Node with an M-DUK containing the same Start Node, and every Mid Node by a C-DUK composed in part by the Mid Node and subtracting the same End Node that is added to the DUK before it. The last C-DUK must also contain the End Node of the path in question.

3.4 DUKs in Cyclone LAB

Fig. 7 shows how DUKs map to LE i and j in a Cyclone LAB. Similar to the Start Node, the M-DUK spans two LEs. Since there are 16 LEs in a LAB, and two input sets (Sec. 2.2), there are $16 \times 15 \times 2 = 480$ M-DUKs. An equal number of C-DUKs exist, since a C-DUK also spans two LEs. Using the 960 DUKs in a LAB, it is possible to

represent any path in the LAB originally represented by a set of LC Nodes. Under Fig. 7 appear two LC Node equations leading to the corresponding DUKs. A subscript prefix on both the LC Nodes and the DUKs indicate the source LE and a subscript suffix signals the sink LE. We can establish a one-to-one correspondence between Start Nodes and M-DUKs (Fig. 7a) by observing that the prefix and suffix on the Start Node matches the prefix and suffix of the M-DUK. Essentially, it indicates that if the Start Node begins in LE i and ends in LE j, the M-DUK will as well. A similar bijection exists between Mid Nodes and C-DUKs (Fig. 7b). The equations in Fig. 7 also indicate which End Nodes must be added or subtracted to correctly form the DUK.

These equations and this notation allows us to trivially transform a path based on LC Nodes into one using DUKs. We replace the Start Node with the M-DUK that has the same source and sink LE. Similarly we replace every Mid Node with the matching C-DUK. The delay contributed by the End Node will already form part of the last DUK. An example will help solidify this transformation.

Consider the path with four LC Nodes

$$ {}_iS_j + {}_jM_k + {}_kM_l + {}_lE$$

Applying the transformation algorithm described above leads to the path

$$ {}_iMD_j + {}_jCD_k + {}_kCD_l$$

Expand each DUK to its LC Node representation leads to

$$\underbrace{{}_iS_j + {}_jE}_{{}_iMD_j} + \underbrace{{}_jM_k + {}_kE - {}_jE}_{{}_jCD_k} + \underbrace{{}_kM_l + {}_lE - {}_kE}_{{}_kCD_l}$$

Which, after simplifying the terms, equals the original LC Node-based path.

It is not a coincidence that the number of DUKs, 960, matches the rank of the matrix formed by paths × LC Nodes. The algorithm above shows how a linear combination of DUKs can be used to represent an arbitrary measurable path. This is the definition of a basis for the matrix. Therefore, these DUKs form a basis for the path-LC Node matrix. As such, by obtaining the delay of the 960 DUKs, we can compute the delay of any of the 10^{18} paths in the LAB.

This basis is superior to the one suggested at the end of Sec. 3.2, where 960 linearly independent paths are selected to form the basis, for several reasons. First, DUKs can be composed incrementally, allowing routing algorithms to easily incorporate this delay information into their path search. Second, DUKs provide a uniformity that the other basis lacks. There is no guarantee that all paths in the other basis will be of the same length or use similar LUT inputs. Therefore, it is not easy to compare delays between and within LABs. DUKs, on the other hand, have two consistent forms, M-DUKs and C-DUKs. We can directly compare one C-DUK using LUT input A to another C-DUK using LUT input A, and know that if one is faster, it is due to process variation and not because of differences in what they represent. Finally, DUKs provide very fine-grain delay information, almost on the order of one LE, while the other basis only has delays of paths.

3.5 Obtaining DUK Delays

It should come as no surprise that it is impossible to measure C-DUKs directly, since one term subtracts the delay of an End Node. It is relatively simple, however, to figure out which paths combine to give a C-DUK's delay. Consider C-DUK ${}_iM_j + {}_jE - {}_iE$ from Fig. 7b. To get this delay we simply measure a path starting with a set of Nodes represented by path prefix A and ending in Nodes ${}_iM_j + {}_jE$ and subtract from it a path starting with the Nodes in A and ending in Node ${}_iE$. This leads to the path equation:

$$(A + {}_iM_j + {}_jE) \; - \; (A + {}_iE) = {}_iM_j + {}_jE - {}_iE$$

In a sense, this mathematically demonstrates the purpose of a C-DUK, removing the last End Node in a path and replacing it with a new Mid Node and End Node.

Since every M-DUK represents the delay of a Start Node plus an End Node and a path must begin at a Start Node and end at an End Node, our path measurement technique (Sec. 2.3) should allow us to directly measure the delay of every M-DUK. Unfortunately, as established in Sec. 2.3, the shortest path we can confidently measure is of length 6, while an M-DUK forms a much smaller path of length 1 LUT and 2 registers (Fig. 7a). Therefore, we take an indirect approach to measuring the delay of an M-DUK by measuring three paths and taking a linear combination of these paths.

To compute the delay of M-DUK ${}_iS_j + {}_jE$, we measure one path that begins by a set of nodes represented by A and ends with ${}_lM_j + {}_jE$. Then measure a second path which begins with ${}_iS_j + {}_jM_k$ and ends with a set of nodes represented by B. Finally we measure a path which is similar to the second path at the beginning and similar to the first path at the end: $A + {}_lM_j + {}_jM_k + B$. Adding the first two paths and subtracting the third leads to the delay of the M-DUK as shown in the following path equation:

$$(A+{}_lM_j+{}_jE)+({}_iS_j+{}_jM_k+B)-(A+{}_lM_j+{}_jM_k+B)={}_iS_j+{}_jE$$

There exist a few requirements on which nodes may form part of A and B. Since the third path uses both A and B, we must make sure that each of the 16 LUTs in the LAB is used only once between the Nodes in A, B, and the two Mid Nodes ${}_lM_j + {}_jM_k$. Also, A and B should not use the LUT i or j. These requirements are easy to satisfy and allow for long paths that we can measure using the limited frequency resources in the Cyclone III and Cyclone IV FPGAs.

All told, we measure two paths for every C-DUK and three for each M-DUK, at worst, this means we must measure $2 \times 480 + 3 \times 480 = 2,400$ paths per LAB. Although this is slightly larger than the minimum of 960 given by performing Gaussian Elimination on the path × LC Node matrix, it is still a small number compared to the total possible paths, and it meets the Timing Extraction goals: Fine-grain measurements suitable for direct variation characterization and component-specific routing.

4. EXPERIMENTAL RESULTS

We applied Timing Extraction both to 18 Arrow BeMicro boards which have a Cyclone III FPGA EP3C16F256C8N [2] and one Terasic DE0-Nano with a Cyclone IV FPGA EP4CE22F17C6N [16], modified to allow control over the FPGA's internal V_{dd}. In this section we present the main results from our measurement experiments on both boards.

4.1 Methodology

The delay of a path in an FPGA is subject to many sources of variation beyond process variation. These include effects such as CAD tool decisions, local supply voltage IR-drop, crosstalk and temperature fluctuations. To annul the effects of these variation sources we perform our measurements in a very structured and systematic way. We divide the FPGA into a control region, where logic required to control the

measurement tests is placed on 66 LABs, and a measurement region containing the LABs that will be measured. This keeps the control logic away from the paths under test so that noise effects in the control circuitry will have minimal impact on the measured circuitry. Leveraging the constraints provided by QUIP [1], the placement and routing of all but the LABs being measured is fixed and consistent for all our measurements. This assures us that signal path lengths and compositions are identical across test and do not directly contribute to the differences in measured delays. QUIP is also used to dictate the placement and routing of the path being measured within a LAB. Moreover, to reduce the overall activity in the FPGA, we do not measure LABs in parallel, but rather measure LABs one at a time. This guarantees that local heating and switching-activity-dependent IR drop do not impact the delay measurements. What's more, all measurements are taken in a temperature controlled room, and we perform our measurement several times to reach a stable internal temperature before recording the final path delay. All these precautions lead to path delays measured in a consistent and precise manner with repeatable results, suggesting the measurements reveal the underlying process variation and allowing us to compare results between LABs and FPGAs without worry that other variation effects cloud our results.

We use the path measurement technique (Sec. 2.3) on 18 Cyclone III FPGAs, to measure the 2,400 paths per LAB necessary to compute all DUK delays. Each measurement set taking on average 20 minutes per LAB. Due to limitations in the Cyclone III PLLs, for our measurements, we increment the frequency at linear intervals of 1.6 ps and at each frequency, perform 2^{15} path measurements, taking as the delay of the path the frequency that yields a 50% failure rate for that path. Unless otherwise specified, throughout this section we present results related to C-DUKs in LAB (27,22) of a Cyclone III. Where appropriate, we indicate more general results.

4.2 Extracted Characterization

Fig. 8 shows the resulting distribution of the paths measured to compute C-DUKs in a LAB. We highlight four separate distributions to isolate two sources of known systematic difference, the path length (7 and 8 LUTs) and the LUT inputs used (A&B or C&D). From these paths, we compute DUK delays, Fig. 9 shows these distributions for C-DUKs in a LAB. In this case, the different colors indicate the LUT input used by the DUK. Fig. 10 shows the individual delays for each C-DUK over LUT inputs A and B. Note that there is no single delay associated with a LUT; each source-sink pair has a unique delay, demonstrating the importance of accounting for LUT to LUT routing. Within a LAB, on average, over all 18 FPGAs we see a standard deviation of $\sigma/\mu = 3\%$ for M-DUKs and $\sigma/\mu = 5\%$ for C-DUKs.

Fig. 11a and 11b compare the C-DUK delay distribution of two LABs in one FPGA, and of one LAB in two FPGAs, respectively. The results indicate that the variation is composed of a spatially correlated component, a within-die correlated component, and a random component. If the variation was only correlated, the data points on these graphs would lie on the $\Delta 0ps$ diagonal line. Similarly, if it was all random variation, the data points would resemble Fig. 1c. The correlated components are less apparent, but random variation is clear when reviewing Fig. 12 which compares the

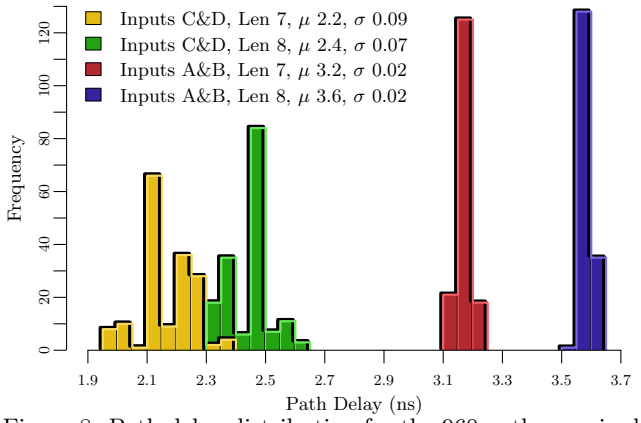


Figure 8: Path delay distribution for the 960 paths required to solve all C-DUKs, differentiating known systematic variation, Cyclone III LAB (27,22)

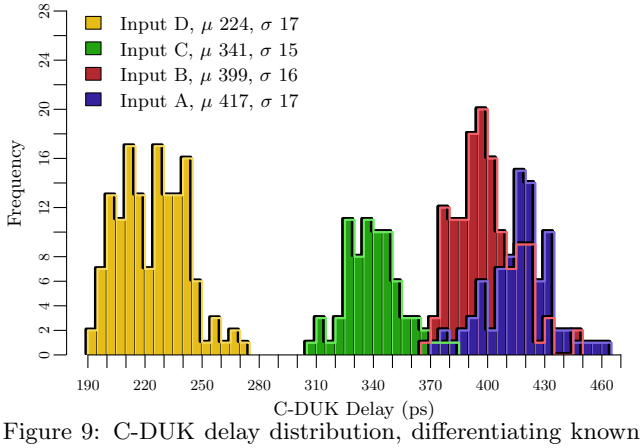
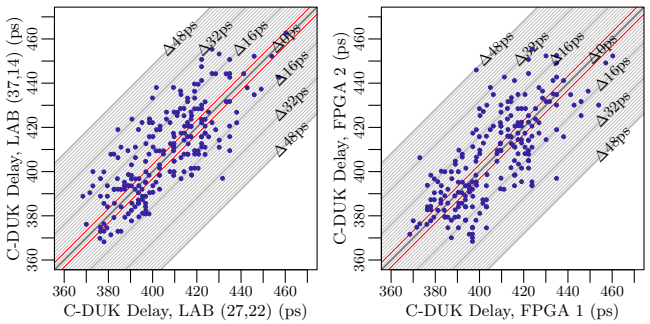


Figure 9: C-DUK delay distribution, differentiating known systematic variation, Cyclone III LAB (27,22)

End LE

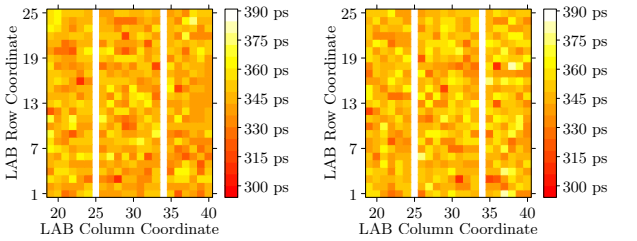
Start LE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		409	384	381	377	389	390	393	393	379	392	395	379	403	382	398
2	400		433	434	403	414	412	420	422	403	417	418	404	422	409	401
3	378	414		409	376	392	395	393	398	382	396	395	379	400	387	376
4	418	417	417		426	434	423	417	420	404	414	431	408	434	409	416
5	385	396	395	419		422	397	401	404	387	399	404	388	407	393	382
6	415	422	423	422	404		453	442	423	406	415	430	411	434	409	415
7	392	388	393	395	373	412		409	390	373	385	398	377	403	377	384
8	396	407	400	401	396	406	409		434	422	409	412	396	417	400	390
9	376	387	387	382	376	387	390	409		392	384	389	370	396	376	368
10	407	419	417	412	411	425	419	423	422		449	445	414	423	409	403
11	379	390	390	385	380	390	393	396	393	414		422	382	396	382	372
12	422	420	422	422	401	418	426	415	417	399	414		426	445	398	409
13	381	388	393	384	379	389	396	398	390	376	396	416		416	377	371
14	417	418	422	412	401	415	428	414	417	398	409	420	399		423	430
15	385	396	401	393	385	398	403	403	403	392	396	401	384	423		398
16	457	460	438	434	414	433	441	430	428	411	425	434	420	438	412	

Figure 10: C-DUK delays in picoseconds over LUT inputs A and B. Rows index start LE of C-DUK, columns index end LE. LUT input A shown by highlighted row header, B otherwise. Cyclone III LAB (27,22)



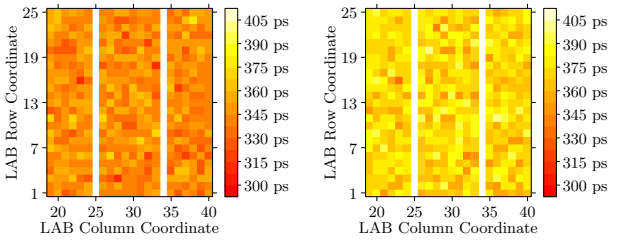
(a) LAB vs LAB, same FPGA (b) FPGA vs FPGA, same LAB

Figure 11: Correlation between C-DUKs in two LABs in one FPGA (a) and between two FPGAs for the same LAB (27,22) (b). Diagonal lines indicate difference between results in terms of $d_\Delta = 1.6$ ps. Thicker lines indicate $10d_\Delta$. Red lines at $\pm 2d_\Delta$ region. Cyclone III



(a) FPGA 1 (b) FPGA 2

Figure 12: Delay heatmap for the C-DUK that goes from LE 10 to LE 8, over a region of 21×25 LABs for two different FPGAs. White columns represent location of embedded blocks. Cyclone III

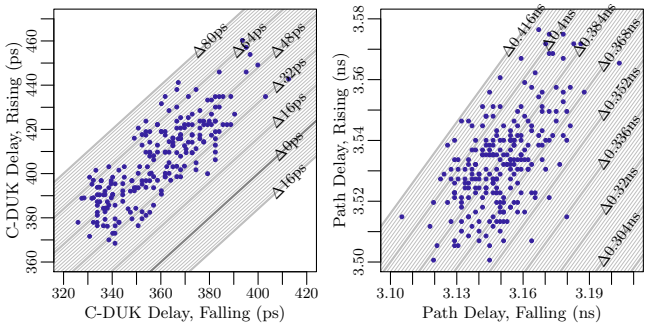


(a) C-DUK $_{10}CD_8$ (b) C-DUK $_{11}CD_8$

Figure 13: Delay heatmap for the C-DUK that goes from LE 10 to LE 8 (a) and C-DUK from LE 11 to LE 8 (b), over a region of 21×25 LABs for the same FPGA. White columns represent location of embedded blocks. Figs. 12a and 13a show same C-DUK using different heat scales. Cyclone III

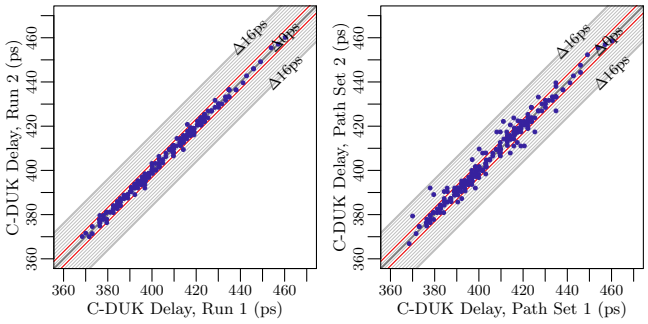
delay of the same C-DUK over a region of 21×25 LABs between two FPGAs. Fig. 13, which compares two C-DUKs in one FPGA over the same region, does show correlated variation, where one C-DUK is clearly slower than the other; however, there still exists a strong random component.

We also see strong evidence of a mixture of variation types when considering the DUK delays for rising transitions as compared to falling transitions (Fig. 14). As previously pointed out, the nature of CMOS and the use of NMOS pass transistors in the FPGA lead us to expect a difference in the delay of rising and falling transitions. On average, falling transitions are 9% faster. However, the spread in Fig. 14a shows a strong random component, due to the fact that



(a) DUK Delay (b) Path Delay

Figure 14: Correlation between Rising and Falling delays for C-DUKs (a) and paths (b). Diagonal lines indicate difference between results. Cyclone III LAB (27,22)



(a) Run vs Run (b) Path Set vs Path Set

Figure 15: Correlation between C-DUKs when measuring the same paths twice (a) and measuring different path sets yielding the same DUKs (b). Diagonal lines indicate difference between results in terms of $d_\Delta = 1.6$ ps. Thicker lines indicate $10d_\Delta$. Red lines at $\pm 2d_\Delta$ region. Cyclone III LAB(27,22)

PMOS and NMOS transistors do not have perfectly correlated relative parameters and can vary independent of each other.

4.3 Measurement Validation

The measurement of the delay of a path can be subject to many sources of noise; therefore, we would like to build confidence that we are not measuring that noise but rather the actual delay of paths and DUKs in a consistent manner. As explained in Sec. 4.1, we control as many aspects as possible when performing our measurements. To measure if these controls achieve consistency, we perform the measurements twice by measuring paths, computing all DUK delays and repeating. Fig. 15a shows the resulting C-DUK delay when we measure paths twice. We see high correlation with nearly all DUKs differing by less than ± 3.2 ps (region between red diagonal lines) between the first and second measurement.

A second form of validation comes from the fact that we can measure distinct sets of paths that allow us to compute the delay of the same set of DUKs. Recall from Sec. 3.5 that we need two paths to compute the delay of C-DUKs and three for M-DUKs. These paths have a fixed set of LC Nodes that determine which DUK will be computed from their delays, and a prefix of LC Nodes that do not form part of the final DUK. We can select a different set of LC Nodes to use for the prefix without affecting which DUKs we compute. Fig. 15b shows the resulting C-DUKs when we compute them using two different sets of paths. Considering that the path measurement

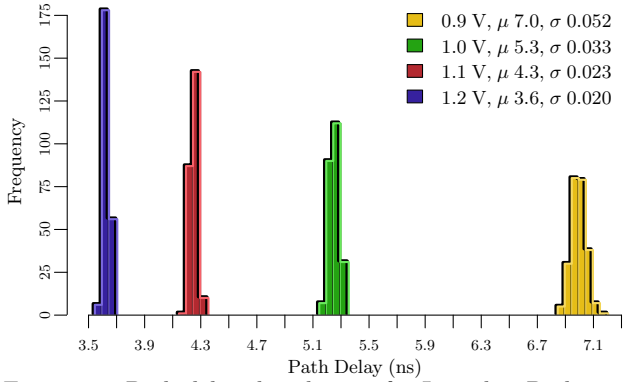


Figure 16: Path delay distribution for Length 8 Paths over LUT inputs A and B required to solve C-DUKs, differentiating varying V_{dd}, Cyclone IV LAB (28,22)

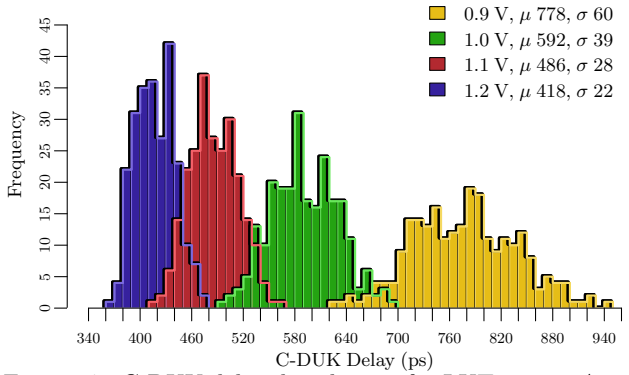


Figure 17: C-DUK delay distribution for LUT inputs A and B, differentiating varying V_{dd}, Cyclone IV LAB (28,22)

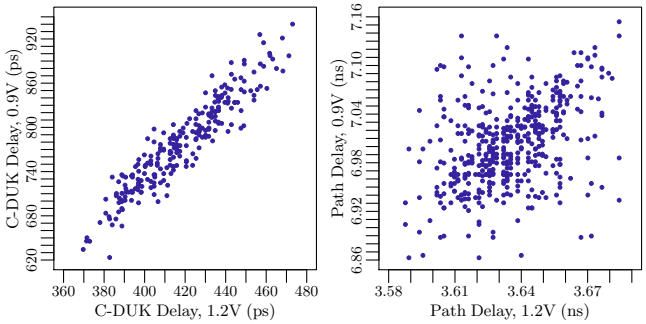
inherently introduces a difference of ±3.2 ps, Fig. 15b shows that it matters little which set of paths are measured as long as we can compute the complete set of DUKs from these paths. Together these figures show that we can trust our technique to correctly and consistently compute the delay of DUKs.

4.4 Effects of Varying V_{DD}

Lowering V_{DD} is a common and important way to save power and energy. In this section we examine the effect that reducing V_{DD} has on variation. In particular we ask whether scaling V_{DD} has a purely systematic effect on the variation distributions or is there a random component as well. To do this we modify a DE0-Nano board containing a Cyclone IV FPGA so that we can control the internal V_{DD}. Nominally, the board provides a 1.2 V V_{DD}. For our tests, we scale at 100 mV increments. At $V_{DD} = 0.8$ V, a large percent of our measurements fail and at 0.7 V the board fails to power up.

We know that a lower V_{DD} increases the propagation delay of a circuit, as well as the standard deviation of the path delay distribution [4]. We clearly see this effect in Fig. 16, the delay distribution for the paths of length 8 used to compute C-DUKs. As we lower V_{DD} the distribution shifts right and becomes wider. This effect is even more pronounced when we look at the C-DUK delay distributions in Fig. 17.

To see how the distribution changes when we go from 1.2 V to 0.9 V we plot correlation graphs (Fig. 18). We would expect a graph similar to Fig. 15a if lowering V_{DD} only had a systematic effect on the distribution. However, we observe a significant random component, indicating that lowering V_{DD} magnifies the impact of random variation.



(a) DUK Delay (b) Path Delay

Figure 18: Correlation between Measuring with V_{dd} at 1.2V and 0.9V for C-DUKs (a) and paths of length 8 (b) for LUT input A and B. Cyclone IV LAB (28,22)

5. GENERALIZING TIMING EXTRACTION

Although Sec. 3 introduces Timing Extraction by applying it to a Cyclone III LAB, the approach generalizes to any FPGA that has registers and configurable PLLs. We can distill the essence of Timing Extraction into five concepts.

1. We can measure the delay of a group of components in the FPGA using only resources already in the FPGA.
2. LC Nodes represent the smallest group of components for which we need to compute a delay, since, if we use any component in an LC Node, we must use all other components in the LC Node.
3. When using the measurement technique from Sec. 2.3, it is not possible to solve for the delay of every LC Node when a measured path begins at a Start Node, goes through zero or more Mid Nodes, and terminates at an End Node.
4. When representing all measurable paths as a matrix, there exists a basis that will allow us to compute the delay of any path in the FPGA using only the delay of vectors in that basis.
5. We can formulate a basis where every vector is a DUK composed of a small linear combination of LC Nodes.

The first, second, and fourth points are immediate; however, it is not obvious why the third and fifth hold true. Although a full explanation, formalization, and proof is beyond the scope of this paper, we can build some intuition to address the third point. Consider a simplified circuit that, when represented in LC Nodes, has all paths being composed by just a Start Node and an End Node. Moreover, there exists a physical path in the circuit formed by combining any Start Node with any End Node. We can represent this situation as a fully connected bipartite graph with Start Nodes forming one set and End Nodes the second. For simplicity, assume that the delay of every path is measured to be 500 ps. It is easy to show that at least two solutions to the delay of the nodes exist. One solution assigns a delay of 200 ps to all Start Nodes and a delay of 300 ps to all End Nodes. The second solution does the opposite, assigning 300 ps to Start Nodes and 200 ps to End Nodes. A similar circuit with fewer paths suffers from the same problem. Therefore, this circuit, and any subset, leads to an underdetermined system. The argument becomes somewhat more complicated when considering the more general problem which also includes Mid Nodes; however, the intuition remains the same.

Showing the fifth point to be true remains part of our future work. We have already introduced two types of DUKs, yet it is likely that more will be necessary to decompose an arbitrary path into DUKs. The exact form and number is not yet clear;

however, we expect that the regularity of FPGAs will help limit the total number of DUK types. By defining enough DUK types to be able to decompose an arbitrary measurable path into DUKs, we will be able to form a DUK basis. Finally, by defining new DUKs also as a small linear combination of LC Nodes, we can keep all DUKs small enough to provide fine-grain, meaningful delay information.

6. FUTURE WORK

The previous section suggests that Timing Extraction is more generally applicable. This paper applies Timing Extraction exclusively to the LABs. To get the full, intended benefits of this technique, it is essential to also apply Timing Extraction to inter-cluster routing and LUT logic. Moreover, the results section hints at the existence of different types of variation: systematic, spatially correlated, random, and shows that Timing Extraction is able to provide the raw information necessary to understand variation in the FPGA. To fully harness the power of Timing Extraction, however, a mathematical analysis of the information it provides should be performed to quantify how much and what kind of variation exists within the FPGA.

Finally, we perform our measurements in a highly controlled setting (Sec. 4.1), this leads to clean and consistent results, yet, it is not clear which controls are necessary for good results. Careful experimentation will reveal how the results change when we change or relax the strong restrictions on our measurement technique, allowing us to simplify and accelerate path measurements.

7. CONCLUSIONS

We presented Timing Extraction, a method used to extract the fine-grained delay information necessary to understand variation within the FPGA and to generate component-specific mappings. We acquire this information using only resources already present in the FPGA. Essentially, we apply a launch and capture technique to measure a subset of all paths in the FPGA, and extract small Discrete Units of Knowledge (DUKs) from these measurements. We can then compose DUKs to compute the delay of any path in the FPGA and use them to understand the amount and type of variation present.

We applied this technique to the Logic Array Blocks in both the Altera Cyclone III and Cyclone IV FPGAs. The results indicate that, on average, we see $\sigma/\mu = 4\%$ variation in the 65 nm process used for the Cyclone III. Moreover, there is clear indication that random variation forms a significant part of the total variation. We expect that as we measure smaller technology nodes, both the total variation and the contribution from random variation will increase. By using Timing Extraction we will be able to characterize and reduce the adverse effects from this increase.

Acknowledgments

This research was funded in part by National Science Foundation grant CCF-0904577. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors gratefully acknowledge donations of software and hardware from Altera Corporation that facilitated this work.

8. REFERENCES

[1] Altera. Quartus II University Interface Program. `http://www.altera.com/education/univ/research/quip/unv-quip.html`.

[2] Arrow. BeMicro FPGA Evaluation Kit. `http://www.arrownac.com/offers/altera-corporation/bemicro/`.

[3] W. B. Culbertson, R. Amerson, R. Carter, P. Kuekes, and G. Snider. Defect tolerance on the TERAMAC custom computer. In *FCCM*, pages 116–123, April 1997.

[4] M. Eisele, J. Berthold, D. Schmitt-Landsiedel, and R. Mahnkopf. The impact of intra-die device parameter variations on path delays and on the design for yield of low voltage digital circuits. *IEEE Trans. VLSI Syst.*, 5(4):360–368, Dec. 1997.

[5] B. Gojman, N. Mehta, R. Rubin, and A. DeHon. Component-specific mapping for low-power operation in the presence of variation and aging. In *Low-Power Variation-Tolerant Design in Nanometer Silicon*, chapter 12, pages 381–432. Springer, 2011.

[6] S. Hanson, B. Zhai, K. Bernstein, D. Blaauw, A. Bryant, L. Chang, K. K. Das, W. Haensch, E. J. Nowak, and D. M. Sylvester. Ultralow-voltage, minimum-energy CMOS. *IBM J. Res. and Dev.*, 50(4–5):469–490, July/September 2006.

[7] X. Li, J. Tong, and J. Mao. Temperature-dependent device behavior in advanced CMOS technologies. In *ISSSE*, volume 2, pages 1–4, Sept. 2010.

[8] M. Majzoobi, E. Dyer, A. Elnably, and F. Koushanfar. Rapid FPGA delay characterization using clock synthesis and sparse sampling. In *Proc. Intl. Test Conf.*, 2010.

[9] L. McMurchie and C. Ebeling. PathFinder: A negotiation-based performance-driven router for FPGAs. In *FPGA*, pages 111–117, 1995.

[10] N. Mehta, R. Rubin, and A. DeHon. Limit study of energy & delay benefits of component-specific routing. In *FPGA*, pages 97–106, 2012.

[11] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits*. Prentice Hall, 2nd edition, 1999.

[12] P. Sedcole, J. S. Wong, and P. Y. K. Cheung. Modelling and compensating for clock skew variability in FPGAs. *ICFPT*, pages 217–224, 2008.

[13] J. R. Smith and X. Tian. High-resolution delay testing of interconnect paths in Field-Programmable Gate Arrays. *IEEE Trans. Instrum. Meas.*, 58(1):187–195, 2009.

[14] A. Srivastava, D. Sylvester, and D. Blaauw. *Statistical Analysis and Optimization for VLSI: Timing and Power*. Integrated Circuits and Systems. Springer, 2005.

[15] E. A. Stott, J. S. J. Wong, P. Pete Sedcole, and P. Y. K. Cheung. Degradation in FPGAs: measurement and modelling. In *FPGA*, page 229, 2010.

[16] Terasic. DE0-Nano Development and Education Board. `http://www.terasic.com.tw/cgi-bin/page/archive.pl?\Language=English&CategoryNo=139&No=593`.

[17] T. Tuan, A. Lesea, C. Kingsley, and S. Trimberger. Analysis of within-die process variation in 65nm FPGAs. In *ISQED*, pages 1–5, March 2011.

[18] J. S. Wong, P. Sedcole, and P. Y. K. Cheung. Self-measurement of combinatorial circuit delays in FPGAs. *ACM Tr. Reconfig. Tech. and Sys.*, 2(2):1–22, June 2009.

[19] Y. Ye, S. Gummalla, C.-C. Wang, C. Chakrabarti, and Y. Cao. Random variability modeling and its impact on scaled CMOS circuits. *J. Comput. Electron.*, 9(3-4):108–113, Dec. 2010.

[20] H. Yu, Q. Xu, and P. H. Leong. Fine-grained characterization of process variation in FPGAs. In *ICFPT*, pages 138–145, 2010.