Rapid FPGA Delay Characterization Using Clock Synthesis and Sparse Sampling

Mehrdad Majzoobi, Eva Dyer, Ahmed Elnably, and Farinaz Koushanfar
Electrical and Computer Engineering Department
Rice University, Houston, Texas 77005
Email: {mehrdad.majzoobi, e.dyer, ahmed.elnably, farinaz}@rice.edu

***Abstract*—This paper introduces a set of novel techniques for rapid post-silicon characterization of FPGA timing variability. The existing built-in self-test (BIST) methods work by incrementing the clock frequency until timing failures occur within the combinational circuit-under-test (CUT). A standing challenge for industrial adoption of post-silicon device profiling by this method is the time required for the characterization process. To perform rapid and accurate delay characterization, we introduce a number of techniques to rapidly scan the CUTs while changing the clock frequency using off-chip and on-chip clock synthesis modules. We next find a compact parametric representation of the CUT timing failure probability. Using this representation, the minimum number of frequency samples is determined to accurately estimate the delay for each CUT within the 2D FPGA array. After that, we exploit the spatial correlation of the delays across the FPGA die to measure a small subset of CUT delays from an array of CUTs and recover the remaining entries with high accuracy. Our implementation and evaluations on Xilinx Virtex 5 FPGA demonstrate that the combination of the new techniques reduces the characterization timing overhead by at least three orders of magnitude while simultaneously reducing the required storage requirements.**

I. Introduction

Scaling of CMOS to miniature feature sizes has provided continuous advantages to digital integrated circuits by increasing the device densities per unit area and reducing the cost per computing function. Fabrication and operation of these minuscule devices has become a big challenge. This is because at the small scale, as devices reach the physical limits of silicon, the fabrication technology faces increased complexity. Due to the uncertainty in the manufacturing process present in the state-of-the-art silicon technologies, transistor properties and other device parameters are no longer deterministic. The variations are likely to increase for the smaller-scale pending technologies.

In conventional designs, yield loss mainly comes from the following two sources: (a) *functional* yield loss because of processing defects, and (b) *timing* yield loss because of timing failures due to processing parameter variations. In current FPGA technologies, the path timings are becoming increasingly volatile, due to variations on path delay and clock skew in manufactured chips. One may decrease the timing yield loss by reducing the circuit sensitivity to process variations and conservative timing, but then the speed improvements in the underlying process technologies are not fully exploited. Addition of pre-silicon timing yield constraints and statistical timing models are of limited help, as the exact post-silicon characteristics have variability around their expected value.

Fortunately, a body of recent work has demonstrated that the timing variations on FPGAs can be characterized and tuned for. To measure the timing performance of components on FPGA, methods for configuring each device to include Built-In-Self-Test (BIST) circuitry were developed [1–3]. Post-silicon optimization and planning methods and tools are being developed for exploiting the fluctuations in process parameters to adjust the circuit performance in accordance with the specific characteristics of each FPGA device [4]. Furthermore, post-silicon tuning has been demonstrated to be effective in practice, improving the timing yield of FPGA circuits [4, 5].

To perform post-silicon characterization, especially in fabrication lines where tens of thousands of ICs are fabricated and shipped daily, the overhead required for testing and timing characterization can be prohibitive. In addition to requiring that the characterization process is fast, the process must also be performed with minimal effort.

In this paper, we introduce novel methods for rapid characterization of FPGA timings at the Look-up Table (LUT) level, significantly improving the test times compared to the available techniques, while maintaining high accuracy. Our contributions are as follows:

- We introduce several methods for on-chip and off-chip clock synthesis to rapidly change the clock frequency. We then combine the clock synthesis methods with three distinct architectures to efficiently characterize an array of CUTs. The advantages and drawbacks of each method are compared in detail.
- The timing failure rate signal from each CUT is analyzed and a compact parametric representation is introduced.
- We develop an efficient strategy to estimate the compact timing parameters of the CUT with as minimal number of measurements at distinct frequencies.
- We show that timings from a array of CUT form a low-rank matrix due to the spatial correlation of the delays across the FPGA die.
- Our rapid characterization method exploits this low-rank property to recover all delays from a subset of CUT delays in the array with high accuracy.

- We provide a working implementation of the newly proposed techniques on a group of Virtex 5 FPGAs. Our evaluation results demonstrate the effectiveness of the new rapid characterization method in reducing the test time by about three orders of magnitude.

The organization of this paper is as follows. After reviewing the related literature and methods on test and characterization of timing variability in Section II, we provide some background on manufacturing process variation and matrix completion using rank minimization and its relevance to the theory of sparse random sampling in Section III. Next, we will describe the details of the timing extraction circuitry in Section IV. In Section V, we present two on-chip and off-chip clock synthesis techniques to determine the most efficient way to extract timing information. Section VI describes the characterization system used to scan and read out information from a 2D array of CUTs on FPGA. In Section VII, a compact parametric representation for the probability of CUT timing failure is presented and utilized to determine the minimum number of samples required for accurate estimation of each CUT delay. In Section VIII, we study the spatial correlation of delay variability across an array of CUTs and exploit this attribute to recover the delays of all CUTs within the 2D array with only a small subset of the CUT delay measurements. Finally, we show a working implementation of these techniques on Virtex 5 chips. We conclude the paper in Section X.

II. Related work

In the past few years, a number of post-silicon FPGA delay characterization methods were proposed. The earlier work focused on using arrays of ring oscillators [6]. The ring oscillators are interfaced with counters that meter the operating frequency of a set of configurable logic blocks. The limitations of this method include the low granularity in timing measurements and the inability of characterizing the impact of delay in clock networks. Testing methods for FPGA delay faults are not proper for delay measurement of nonfaulty devices. The goal in FPGA delay fault testing is to determine whether devices, including interconnects and logic components, have outstandingly larger delays than a certain threshold [1, 7, 8]. Application-specific tests for FPGA characterization are available but they cannot be directly used for a general delay measurement [5]. Methods for profiling the FPGA clock variability based on differential delay measurement circuitry were developed [2, 3]. However, these methods exhibit a major limitation; the test time for a large number CUTs takes a very long time. The characterization results can be used for variability compensation, post-silicon tuning, and variation-aware place and route to improve performance and yield [2, 4, 9, 10].

The BIST structure employed in this paper was developed in [1, 3, 11–13] while other BIST methods for FPGA characterization are also available [14]. The concepts behind our rapid characterization method are generally applicable to other BIST methods, but the implementation details would be specific to each platform. To the best of our knowledge, this is the first work that addresses the time complexity and accuracy of the delay characterization and presents novel methods for performing rapid characterization. In addition, we address the trade off between characterization time and accuracy.

III. Preliminary

This section presents preliminaries on manufacturing process variation, the use of matrix rank minimization for matrix completion, and its connection with sparse random sampling.

A. Process variation

Process variations are the fluctuations in the device parameters and characteristics caused by imperfections and uncertainties in the fabrication process. The total process variation can be viewed as the sum of inter-die and intra-die variations. Inter-die variations refer to differences among the devices on various dies and are constant within one die. Intra-die variations, on the other hand, account for the differences among devices on the same die. The intra-die component can be further divided into spatially correlated and uncorrelated random components. The uncorrelated random variations are primarily caused by the fundamental intrinsic atomic-scale randomness of the materials that make each device, while systematic correlated components stem from unintentional shifts in processing conditions such as mask errors, lithographic off-axis focusing, and reticle stepper alignment errors [15]. Thus, the process variation can be represented by Equation 1, where P_{nom} is the nominal parameter value, ΔP_{inter}, is the inter-die variation component, and $\Delta P_{spatial}(x_i, y_i)$ and $\Delta P_{rand}(i)$ are the spatially correlated and random components of intra-die variation for device i respectively [16].

$$P = P_{nom} + \Delta P_{inter} + \Delta P_{spatial}(x_i, y_i) + \Delta P_{rand}(i) \quad (1)$$

The spatially correlated component of variation can be modeled as the sum of systematic variations and correlated random variations [17]. In CMOS technology, threshold voltage (V_{TH}) variations due to random dopant fluctuation (RDF) make up a considerable portion of the random component. Variations in thin film thickness (T_{ox}), line-edge roughness, gate length and width (L_{eff}, W_{eff}) demonstrate spatial correlations [15]. In this paper, we take advantage of the spatial correlation present in process parameters to perform sparse random sampling over the FPGA die to more efficiently extract the variation information.

B. Sparse random sampling and matrix rank minimization

In traditional Nyquist sampling schemes, a signal of interest must be sampled at a rate twice that of its highest frequency component to perfectly reconstruct the signal from the samples. An emerging field, known as *compressive sampling*, shatters the Nyquist limit and enables sub-Nyquist sampling rates when signals only occupy a small portion of the total bandwidth.

Porting this simple example into a more general framework, the total number of samples required to properly represent a signal that is sparse in an arbitrary basis, is linear in sparsity in the signal (the number of non-zero coefficients used to represent the signal) [18] and logarithmic in signal length. In other words, for very sparse signals, the number of required samples can be much less than that required by Nyquist theory.

Compressive sampling theory may be extended to other low-dimensional signal models, such as manifold models [19] and matrices that exhibit low-rank structure [20]. For low-rank matrices, one may show that by sampling only a small subset of the total entries in the matrix, the remaining entries may be recovered with high accuracy [20]. In concordance with compressive sampling theory, the number of sampled entries scales linearly with the rank of the matrix. In Section VIII, we will show how these results may be utilized in the context of delay characterization across a FPGA array, namely by sparse random sampling of entries from the entire delay array and recovering the missing entries via a nuclear norm minimization problem.

IV. Delay extraction

To measure the delays of logic components inside FPGA, the circuit shown in Figure 1 can be used as suggested in [1, 3, 21]. The target component/interconnect delay to be extracted is called the *Circuit Under Test (CUT)*. Three flip flops (FFs) are used in this delay extraction circuit: *launch FF*, *sample FF*, and *capture FF*. The clock signal is routed to all three FFs as shown on the figure.

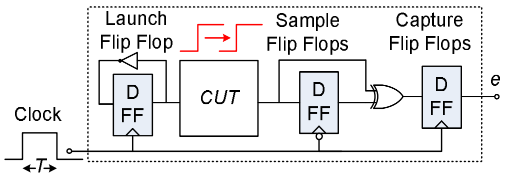


Figure 1. Delay characterization circuit.

Assuming the FFs in Figure 1 were originally initialized to zero, a low-to-high signal is sent through the CUT by the launch FF at the rising edge of the clock. The output is sampled T seconds later on the falling edge of the clock. If the signal arrives at the sample flip flop before sampling takes place, the correct signal value would be sampled, otherwise the sampled value would be different indicating a timing error. The actual signal value and the sampled value are compared by an XOR logic and the result will be held for one clock cycle by the capture FF.

A more careful timing analysis of the circuit reveals the relationship between the delay of the CUT (t_{CUT}), the clock pulse width (T), the clock-to-Q delay at the launch FF t_{clk2Q}, and the clock skew between the launch and sample FFs t_{skew}. The setup/hold of the sampling and capture FFs are denoted by t_{setS}, t_{holdS}, t_{setC}, and t_{holdC} respectively. The time it takes for the signal to propagate through CUT and reach the sample flip flop from the moment the launch flip flop is clocked is represented by t_P. Based on the circuit functionality in Figure 1,

$$t_P = t_{CUT} + t_{clk2Q} - t_{skew}. \tag{2}$$

Therefore, if there are no timing errors in the circuit, the following relationships must hold:

$$t_{holdC} < t_P < T - t_{setS} \tag{3}$$

The errors start to appear if t_p enters the following interval:

$$T - t_{setS} < t_P < T + t_{holdS} \tag{4}$$

The flip flops enter a metastable operation because of the setup and hold time violations and the sampled value becomes nondeterministic. The probability that the metastable state resolves to a 0 or 1 is a function of how close T is to t_P. For instance, if T and t_{CUT} are equal, the signal and the clock simultaneously arrive at the sample flip flop and the circuit would produce an error 50% of time. The probability of observing timing error increases as t_p gets closer to the upper limit of Inequality 4. If the following condition holds, then timing error happens every clock cycle:

$$T + t_{holdS} < t_P < 2T - (t_{setC} + t_{XOR}) \tag{5}$$

Without loss of generality, in our implementation each CUT consists of 4 LUTs (where each implement an inverter). The circuit in Figure 1 is pushed into two slices (one CLB). The CUT delay can roughly be a representative delay for the FPGA slice where it belongs. We refer to the circuit in Figure 1 as *characterization cell* or simply *cell* in the remainder of the paper.

V. Clock Synthesis

Changing the clock's pulse width is a precursor for our characterization. Thus, finding efficient methods to change the clock pulse width could yield significant savings in delay characterization. In this section, we discuss two methods for clock generation using on-chip and off-chip components. We then explain how each method can be integrated into the characterization system.

A. *Linear frequency sweep with off-chip components*

Using off-the-shelf and inexpensive function generators, one can easily perform a linear sweep from a low frequency (f_{min}) to a high frequency (f_{max}) in a continuous fashion (see Figure 3 (a)). Notice that in practice, there is often a minimum recovery time between each sweep (denoted by T_{trans}). Although many inexpensive generators can not synthesize high frequencies (usually the maximum frequency is the range of 20MHz), using the PLLs inside the FPGA, both f_{min} and f_{max} can be shifted up to any desired range. Here, we first investigate the relation between the number of generated clock pulses and the sweep time, the pulse number and the corresponding pulse width. Later, we will show how to efficiently and quickly characterize a large group of logic components with a single (or a few) linear sweeps. Let us assume that the clock frequency to the system

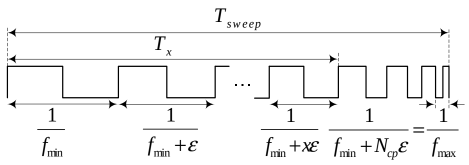

T_{sweep}
T_x
...
$\frac{1}{f_{min}}$
$\frac{1}{f_{min}+\varepsilon}$
$\frac{1}{f_{min}+x\varepsilon}$
$\frac{1}{f_{min}+N_{cp}\varepsilon}=\frac{1}{f_{max}}$

Figure 2. Linear sweeping of the clock frequency.

is swept linearly and continuously in T_{sweep} seconds from $f_{min} = \frac{1}{2T_{max}}$ to $f_{max} = \frac{1}{2T_{min}}$, where $T_{min} < t_p < T_{max}$ (see Figure 2). The relationship between total number of clock pulses in each sweep, N_{cp}, the sweep time, T_{sweep}, initial frequency, f_{min}, and terminal frequency, f_{max} can be derived from Figure 2 and is given in Equation 6:

$$T_{sweep}(N_{cp}) = \sum_{k=0}^{N_{cp}} \frac{1}{f_{min} + k \times \epsilon}, \tag{6}$$

where ϵ is given by

$$f_{max} = N_{cp} \times \epsilon + f_{min}. \tag{7}$$

By replacing ϵ in Equation 6 from Equation 7 and using Riemann method, the sum can be approximated by definite integral as given in Equation 8.

$$\begin{aligned} T_{sweep} &= \sum_{k=0}^{N_{cp}} \frac{1}{f_{min} + \frac{f_{max}-f_{min}}{N_{cp}} \times k} \\ &= \frac{N_{cp}}{f_{max} - f_{min}} \int_{f_{min}}^{f_{max}} \frac{1}{x} dx \\ &= N_{cp} \times K_c \end{aligned} \tag{8}$$

where, $K_c = \frac{\ln(f_{max}) - \ln(f_{min})}{f_{max} - f_{min}}$. The goal is to determine the width of the x-th clock pulse in Figure 1. The time at which the x-th clock width appears during the sweep (T_x) can be determined by Equation 9.

$$T_x = \sum_{k=0}^{x} \frac{1}{f_{min} + \frac{\ln f_{max} - \ln f_{min}}{T_{sweep}} \times k}. \tag{9}$$

By knowing T_x, the clock pulse width associated with the x-th clock cycle can be easily calculated using the following linear interpolation:

$$f_x = (f_{max} - f_{min}) \times \frac{T_x}{T_{sweep}} + f_{min}. \tag{10}$$

In practice, we use a counter to count the clock pulses. The value of the counter is retrieved, every time the output of the characterization circuit is scanned. Then, by using the transformation in Equations 9 and 10, we can easily convert the recorder clock pulse number to the corresponding clock pulse width.

B. *Frequency sweep with on-chip PLLs*

Many modern FPGAs are equipped with built-in clock management modules such as PLLs and DCMs. However, unlike linear sweeping, the frequencies generated with PLLs are discrete. Using a set of PLLs in series, as shown in [1], a large number of frequencies can be synthesized with fine resolution. Dynamic reconfiguration of the PLLs settings (i.e. division and multiplication coefficients) can be used to switch the clock frequency during the operation. However, dynamic reconfiguration of PLLs and DCMs requires putting the circuit to halt state. In addition, as illustrated in Figure 3 (b), changing the PLL frequency incurs a time overhead defined by the frequency/phase lock time. For instance, the PLL nominal lock time in Xilinx Virtex 5 FPGAs is about $100\mu s$ [22]. Therefore, any change in frequency is followed by partial reconfiguration time plus PLL phase lock time. In order to reduce the time overhead, some level of pipelining can be introduced among a group of PLLs such that a PLL whose phase and frequency has already locked is used while another one is still in the locking process.

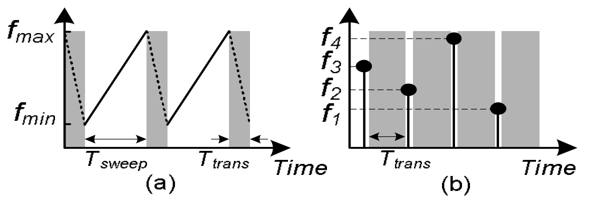


Figure 3. Clock synthesis with (a) linear frequency sweep, and (b) on-chip PLLs.

VI. Characterization System

So far, we have only dealt with characterization of a single CUT and methods for changing the clock frequency. The system shown in Figure 4 can be used to characterize the delays of an array of CUTs on the FPGA. Each cell in the array (square in Figure 4) contains the characterization circuit shown in Figure 1. There is a number of techniques that can be considered for sweeping the clock frequency and scanning the characterization cells. In the following, we present the system architecture for each technique and discuss the implementation and characteristics time overhead for each technique. We investigate the following three characterization techniques:

- linear sweep with serial output scanning,
- using PLL with grouped output scanning,
- linear sweep with grouped output scanning.

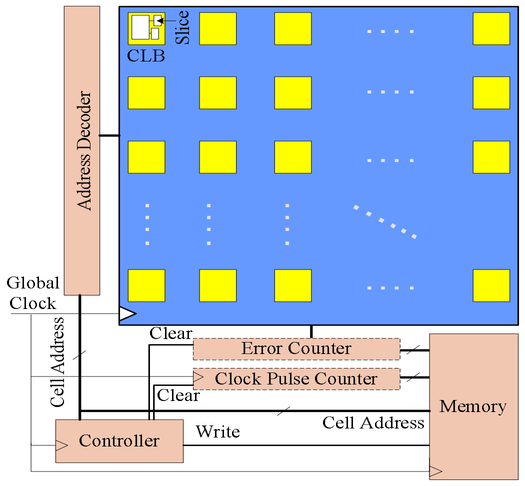


Figure 4. The architecture for chip level delay extraction.

A. Linear sweep with serial output scanning

This technique uses an external function generator to linearly sweep the intended frequency range. The cells are scanned in sequence such that one cell is enabled at a time and undergoes a single sweep. The address decoder points to a new cell in each frequency sweep. The system needs to wait T_{trans}^{linear} seconds (i.e., recovery time) for the clock generator to begin a new sweep. At each distinct clock frequency, the timing errors from each cell are accumulated over 2^w clock pulses using an w-bit pulse counter. If N_f is the desired number of distinct frequency points per cell per sweep, the clock generator should then make $N_f \times 2^w$ clock pulses in each sweep. The total time needed for characterization can be computed by Equation 11, where N_{cells} is the number of cells in the characterization system, and K_c is the coefficient that converts number of pulses to sweep time as defined in Section V.

$$T_{char} = N_{cells} \times \left(N_f \times 2^w \times K_c + T_{trans}^{linear}\right). \qquad (11)$$

Note that since only one cell is being considered for each sweep, the error counter can be shared among all the cells.

B. Using PLL with grouped output scanning

This technique uses the FPGA's internal PLLs to generate N_f pre-calculated distinct frequencies. After tuning the clock to a known frequency, 2^w clock pulses are sent to all the cells at the same time. Each cell contains a w-bit error counter to accumulate the errors. Then, the cells will be disabled to begin writing the error values to the memory. The error values for the characterization cells are read in groups of k cells (k is determined by the number of bits which can be written to the memory in a single clock cycle). Therefore, $\frac{w \times N_{cells}}{k}$ clock cycles are needed to completely save the parameters for a system with N_{cells} cells. The total characterization time for this approach can be calculated using Equation 12, where f_{ave} is the average operating frequency:

$$T_{char} = N_f \times \left(\frac{2^w + \frac{w \times N_{cells}}{k}}{f_{ave}} + T_{trans}^{PLL}\right). \qquad (12)$$

The main problem with this approach is the extra overhead of reconfiguring the PLL plus the lock time (T_{trans}^{PLL}) which is in the order of $100\mu s$ in modern FPGAs [22] (see Figure 3 (b)). Additionally, this solution requires a more complex structure because of the necessary PLL frequency-changing control circuitry.

C. Linear sweep with grouped output scanning

This technique integrates the previous two approaches. The idea is to use an external function generator like the first technique, sweeping all the cells at the same time while reading the error counters from the cells in a grouped manner, like the second technique. This solution does not require extra control circuitry as in the second approach. The main advantage of this approach is that the sweep time can be adjusted so that the system can be characterized in one sweep. Similar to the second case, each cell includes a w-bit error counter to accumulate the number of error occurrences in 2^w clock pulses. Succeeding 2^w clock pulses, the system disables all the cells, i.e., it stops recording new errors. After disabling all characterization cells, we have the error counters inside the cells loaded with the number of errors that occurred in the short burst of clock pulses. With the capability to write $k - bits$ at the same time at each clock cycle to the memory, for a N_{cells} cell system, $\frac{w \times N_{cells}}{k}$ more clock pulses are needed to record all the values of the error counters inside the characterization cells. Thus, the total characterization time for this approach can be calculated using Equation 13:

$$T_{char} = N_f \times \left(2^w + \frac{w \times N_{cells}}{k}\right) \times K_c. \qquad (13)$$

In addition, this technique uses a counter that records the number of clock pulses in one sweep. The value of this counter is recorded every time the characterization system gets disabled. This value can be used to calculate the frequency to which these error counters correspond as explained in Section V.

The characterization time required by the second and third technique is comparatively lower than the first technique. The third technique is much faster than the second PLL based technique, however, its application depends on the availability of off-chip clock synthesizer to perform linear sweeps. In general, the system needs to store $w - bit$ for each cell at each frequency, and a l-bit frequency index from the clock counter/PLL coefficients. The size of the counter should be large enough to avoid overflow when counting the clock pulses in a single sweep. Thus, the total needed memory is $N_f \times (N_{cells} \times w + l)$.

VII. CUT Delay characterization with minimal measurements

In many practical applications, the effort required to collect a full sampling of a signal or phenomenon of interest may be prohibitive. In the case of timing variability characterization of FPGAs, a sweep across different frequency points must be performed in order to extract the delay of each CUT. The number of distinct clock frequencies (N_f) and the clock pulses used at each frequency (2^w) determines the overall amount of time and memory required to characterize each CUT. In this section, we introduce a compact parametric representation for CUT timing failure probability as a function of the input clock frequency. We then use the derived formulation to accurately estimate the CUT delay by trying minimal number of distinct frequecies and clock pulses at each frequency.

The circuit in Figure 1 can be viewed as a system with an analog input (clock pulse width) and a digital binary output. The probability of output being zero or one is a function of the input clock pulse width. It can be shown that this probability can be accurately modeled by a Gaussian CDF whose mean is equal to t_p and standard deviation is related to the setup/hold time of the sample flip flop. The Gaussian nature of the error probabilities can be explained by the central limit theorem [21]. Equation 14 shows the parameterized error probability function:

$$f_{t_p,\sigma_p}(T) = Q(\frac{t_p - T}{\sigma_p}), \tag{14}$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) du$. Our goal is to estimate t_p and σ_p by collecting a set of data points (T_i,e_i); where $e_i \in \{0, 1\}$ is the i-th error flag recorded when the clock pulse width equals T_i; i=1,2,...,M; and M is the total number of measurements. The data points can also be collected for the same clock frequency multiple times. In other words, the input to the cell T_i can be repeated for $r = 2^w \geq 1$ times and averaged to reach a higher resolution at each point. We denote the normalized accumulated error for the input T_i by e_i^r, where $e_i^r \in \{0, 1/r, 2/r, ..., 1\}$. The delay parameters are estimated by least-squares fit:

$$(t_p, \sigma_p) = \text{argmin}||\mathbf{e^r} - f_{t_p,\sigma_p}(\mathbf{T})||_2 \tag{15}$$

The concept is demonstrated in Figure 5, where y-axis shows the CUT timing failure probability and the x-axis represents the input clock width (T). Two sampling scenarios

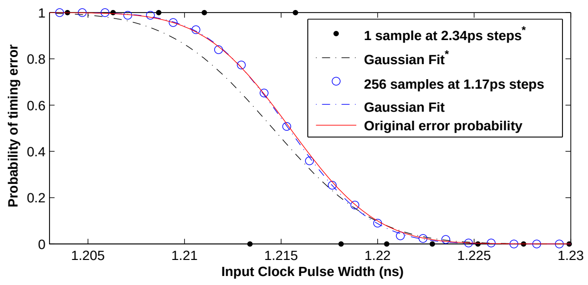


Figure 5. The estimated error signal (y-axis) versus the input clock pulse width (x-axis) reconstructed from (a) 16 samples; and (b) 64 samples.

are compared. In the first scenario, the frequency points are equally spaced at 2.34ps and one clock pulse is sent to the cell at each clock frequency. In the other scenario, the frequency points are equally spaced at 1.17ps and 256 clock pulses are repeated to obtain the accumulated error at each clock frequency. A Gaussian curve is fit to the collected error values. In Figure 5, both the original and the estimated curve are shown on the plots, but they are very close to each other demonstrating a low estimation error while the second scenario takes 512 times longer. We argue here that the estimation accuracy is less sensitive to the number of repetitions than to the number of frequency samples. In the experimental results, we demonstrate that with a fixed budget for total number of tests on the circuit, having more samples in different frequencies results in smaller estimation error than having less frequency samples with higher resolution (repetition) for each sample.

VIII. Minimizing the number of measurements collected across the array

In the previous section, we discussed ways to minimize the number of measurements required to extract the delay of each CUT. In this section, we describe how recent results in rank minimization and matrix completion [20] may be leveraged to minimize the number of CUTs that must be characterized across an array of CUTs, while still enabling recovery of the timing variability across the entire FPGA die.

Let us begin by forming a 2D matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$, where the $\{i, j\}$ entry contains the extracted delay of the CUT inside the $\{i, j\}$ cell (see Figure 4) in the FPGA array. The question we address is whether a sparse sampling of the n^2 entries in this matrix are sufficient to recover the missing entries in the array with high accuracy. Recovering the missing entries from this sparse sampling is an ill-posed problem because there are infinitely many $n \times n$ matrices entries can perfectly agree with the known observed entries. Fortunately, the low rank structure of the timing variability across the array of CUTs can be leveraged to restrict the space of possible solutions. The low rank structure of the

CUT delay array is a consequence of the spatial correlation present in process variation.

The problem of recovering a low rank matrix from a sparse sampling of its entries, was recently studied in [20]. The authors prove that if the number of entries sampled from a $n \times n$ matrix obeys,

$$m \geq C n^{1.2} R \log(n), \tag{16}$$

where C is a constant, a convex optimization routine will recover the underlying rank R matrix with high probability.

In order to determine the number of CUTs that must be sampled from across each FPGA array, we must first show that the delay arrays have low rank. In the next section, we analyze the structure of the singular values of a collection of delay arrays and show that for all of the FPGAs we examined, their corresponding delay matrices are of very low rank. We will then enforce this structure when recovering the delay arrays from a subset of measurements from across the chip.

A. *Analysis of correlations across cells in the delay array*

In this section, we study the spatial correlation of delay measurements for an array of neighboring cells which span the extent of the FPGA fabric. To do this, we use measurement data collected from a set of 12 Xilinx Virtex 5 chips. Strong correlations among the delays in cells, especially in the vertical and horizontal directions, are present in all of the chips we examined. In Figure (6), we show a few examples of the types of spatial structure observed across the entire array of measurements. With a standard tool like principal components analysis (PCA), we can study the structure and correlation in the array more rigorously. To do this, we first find a singular value decomposition of our delay matrix $\mathbf{D} = \mathbf{U\Sigma V^T}$, where $\mathbf{\Sigma} \in \mathbb{R}^{\mathbf{n}\times\mathbf{n}}$ is a diagonal matrix with entries corresponding to the singular values of

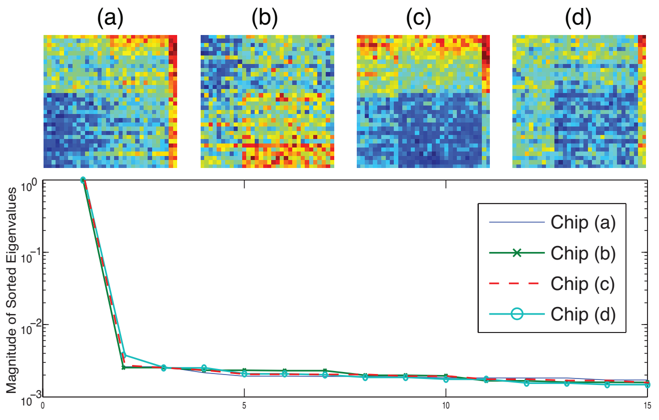


Figure 6. (Top) Examples of delay matrices obtained via our characterization method; (Below) the sorted magnitude of each array's corresponding eigenvalues.

$\mathbf{D}$ in descending order. The columns of $\mathbf{V} \in \mathbb{R}^{\mathbf{n}\times\mathbf{n}}$ form an orthonormal basis that spans our input space and the columns of $\mathbf{U} \in \mathbb{R}^{\mathbf{n}\times\mathbf{n}}$ to form a basis over our output space.

In Figure (6), we show the magnitude of the singular values of $\mathbf{D}$ for delay matrices from four different FPGA devices. The quick decay of the singular values demonstrates that the delay matrices are indeed of low rank.

B. *Recovering delay arrays from incomplete measurements*

Now that we have confirmed that the array of CUT delays are of low rank, we will now discuss how to leverage this structure to recover the timing variability across the FPGA die from only a sparse sampling of the array. After collecting a small subset of measurements from the array, the indices of these entries may be stacked into the index set Ω.

To recover the full delay array, we seek the lowest rank matrix that agrees with all of the observed entries,

$$\min_{\mathbf{A}} \quad \text{rank}(\mathbf{A}) \quad \text{s.t.} \quad \mathbf{A}_{i,j} = \mathbf{D}_{i,j} \quad \forall\{i,j\} \in \Omega, \tag{17}$$

where Ω is an index set containing the indices of all the sampled entries in $\mathbf{D}$, and $\mathbf{A}$ is the low rank matrix that we approximate $\mathbf{D}$ by.

If a sufficient number of measurements is taken in accordance with Equation (16), it can be shown that the following convex relaxation of the rank minimization problem will recover the original array $\mathbf{D}$ exactly.

$$\min_{\mathbf{A}} \quad \|\mathbf{A}\|_\star \quad \text{s.t.} \quad \mathbf{A}_{i,j} = \mathbf{D}_{i,j} \quad \forall\{i,j\} \in \Omega, \tag{18}$$

where $\|\mathbf{A}\|_\star$ is the nuclear norm or trace norm of $\mathbf{A}$. The nuclear norm may also be written as the absolute sum of the singular values of $\mathbf{A}$.

Due to the strong correlations in all of the $\mathbf{D}$ matrices that we examined, solving this nuclear norm minimization problem allows us to recover a low rank matrix $\mathbf{A}$ that effectively characterizes the delay variations across the entire FPGA. We report the typical number of measurements required for this recovery in Section IX-C.

IX. Experimental evaluations

For delay characterization, the system shown in Figure 4 is implemented on the Xilinx Virtex 5 FPGA. The systems contains a 32×32 array of characterization circuitry as shown in Figure 1. A 1-bit counter is placed inside each cell that disables the FFs in the cell after one clock cycle until the counter is reset. The CUT inside the characterization circuit consists of 4 inverters each being implemented using one 6-input LUT. The characterization circuit is pushed into 2 slices (one CLB) on the FPGA. In fact, this is lower limit on number of slices that can be used to implement each characterization circuit. This is because interconnections inside the FPGA forces all the flip flops inside the same slice to use either the clock's rising edge or its falling edge. Since the launch and sample flip flops must operate on different clock edges, they cannot be placed inside the same slices. Figure 7 shows the placement of CUTs on the FPGA. The red line on the figure shows the clock network distribution. Notice that the clock delay skew (t_{skew}) between the launch and sample flip flops is minimal because the branching of the clock interconnection happens at minimum distance to the flip flops. Also, it can be seen the CUTs are pushed into two slices in one CLB.

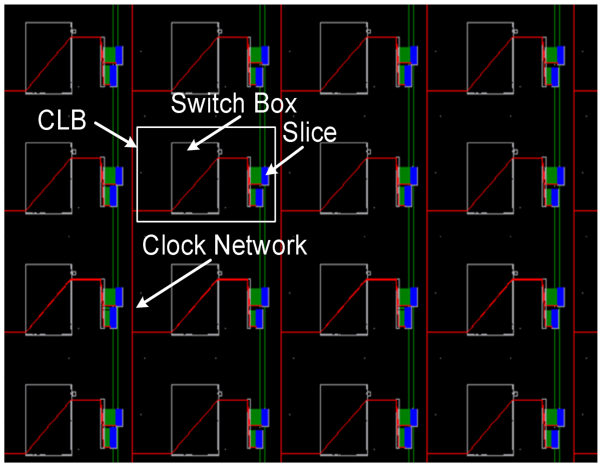
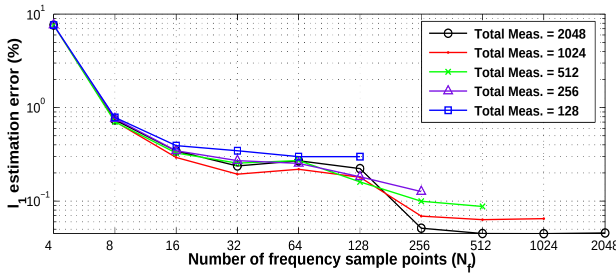


Figure 7. The placement of characterization cells on the FPGA and the clock distribution.

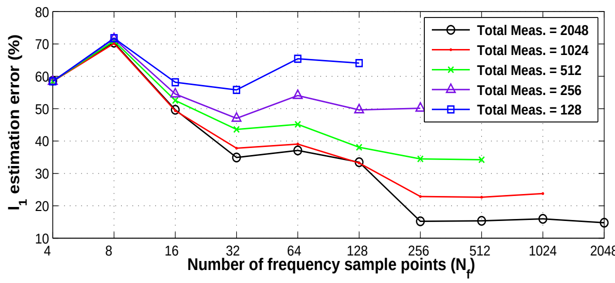
We use an ordinary desktop function generator to sweep the clock frequency from 10MHz to 15MHz and afterwards shift the frequency 34 times up using the PLL inside the FPGA. The measured accumulated error values are stored on a logic analyzer and the data are transferred to computer for further processing. Notice that the storage operation can easily be performed without the logic analyzer by using any off-chip memory. The system is implemented on twelve Xilinx Virtex 5 XC5VLX110 chips. The characterization system in total uses 2048 slices for the characterization circuit array and 100 slices for the control circuit out of 17,280 slices. The experiments were run under normal operational conditions, i.e., room temperature and supply voltage of 1 volts.

A. Sampling rate and accuracy

Next we examine the estimation accuracy of the CUT delay t_p and flip flop transition window width σ_p (see Equations 2 and 14) for a fixed number of total measurements (tests) on each characterization circuit. Since the actual CUT delay values on FPGA is unknown, it is impossible to find the estimation error through measurement. In order to find the estimation error, we simulated the characterization circuit and sampled the simulated system instead. The simulated system follows the behavior expressed by Equation 14. The simulated system takes the measured delay from FPGA as input and reproduce the same indeterministic random behavior as the actual system. Figure 8 (a) shows the estimation error versus the number of frequency points uniformly spaced between T_{min}=1ns and T_{max}=1.4ns. T_{min} and T_{max} are bounds on the smallest and largest delays derived from the distributions in Figure 9 (a) over all chips. The total number of measurements equals the number of frequency points times the number of repetitions for each sample point, i.e., $N_f \times 2^w$. As it can be seen in Figure 8 (a) very low estimation error can be achieved by taking 256 frequency samples and 1 repetition for each sample. Figure 8 (b) shows the same results for σ_p. The estimation error follows the same descending trend as the number of distinct frequency points and repetitions at each frequency are increased.



(a) Estimation error for t_p

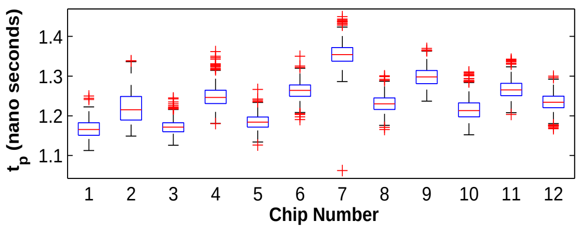


(b) Estimation error for σ_p

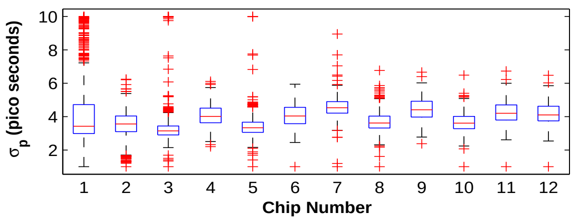
Figure 8. Normalized ℓ_1 estimation error for fixed number of total measurements, and different number of distinct frequency points (N_f) and repetitions (2^w).

B. Delay measurement

Figure 9 (a) and (b) show the distribution of measured propagation delays (t_p) and transition slope (σ_p) for the twelve XC5VLX110 chips. The observed level of timing variability across all the chip has a standard deviation of 5% (of the nominal value). Notice that the transition slope parameter is smaller than 10ps at all chips. The data is collected with high accuracy using 256 frequency points and one repetition for each sample.



(a) Distribution of t_p



(b) Distribution of σ_p

Figure 9. Distribution of the measured delays, t_p (a) and the transition slope, σ_p (b) for all 12 Virtex 5 FPGAs.

C. Recovery of delay arrays

To solve the nuclear norm minimization problem in Equation 18, we used CVX, a general-purpose package for specifying and solving convex programs [23]. Note that we have already performed delay measurement across the full CUT array and use these full samplings to study the performance of our sub-sampling strategy. Next, a subset of the delays in the array are selected randomly to form an incomplete matrix. Then, by using the rank minimization and the fact that the underlying delay matrix is low rank, we attempt to recover the missing entries based on the known entries. In Figure 10, we show the progressive reconstructions of the missing CUT delays for one chip as 20% to 100% of the number of cells are measured. The delays shown next to the colorbar are in nano seconds.

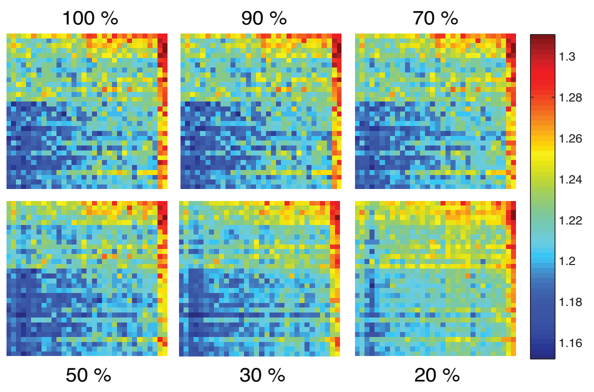


Figure 10. Progressive reconstructions of the delay matrix for a single FPGA.

The recovery error of the missing entries in the array is computed as,

$$e_r = \frac{1}{|\bar{\Omega}|} \sum_{\{i,j\} \in \bar{\Omega}} \frac{|\,\mathbf{A}_{i,j} - \mathbf{D}_{i,j}\,|}{\mathbf{D}_{i,j}}, \tag{19}$$

where e_r is the normalized recovery error, $\bar{\Omega}$ is the set of indices of missing entries, $|\bar{\Omega}|$ is the cardinality of the set $\bar{\Omega}$, or basically the number of missing entries, $\mathbf{A}$ is the recovered delay from solving rank minimization, and $\mathbf{D}$ is actual value of the missing entry. In Figure 11, we show the normalized recovery error of the missing CUT delays versus the percentage of the measured CUT delays for all 12 chips under study. To ensure a reconstruction error of less than 2%, at least 20% of the total CUTs must be measured. The trend of the reconstruction error for all chips suggest that as larger number CUT delays are measured, the reconstruction error of the missing entries will be smaller.

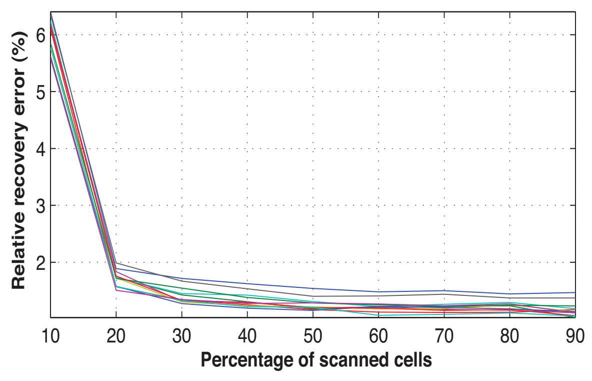


Figure 11. Reconstruction error versus percentage of the measured cells. Each curve corresponds to one of the 12 chips under study.

D. Characterization speed

Table I shows the characterization time and the memory size required to store the data for the 32×32 array in our experiment and also for characterizing the largest Virtex 5 FPGA for both grouped output approach with linear and PLL clock generation techniques. The parallel BIST method proposed in [1] characterizes an array of 52 × 16 cells in 3 seconds and requires $13kbit$ of memory while the results in Table I suggest an improvement of three orders of magnitude in characterization time. Also considering the results in Secion IX-C and by measuring 50% of the cells instead of the whole array, the run times and the memory sizes in Table I will be halved.

X. Conclusions

This paper presented several contributions to rapid characterization of the propagation delays of FPGAs building blocks. We proposed techniques to minimize the time overhead of varying clock frequency during the characterization

Linear Sweep				
N_f	N_{cells}			
	1024		51840	
	T_{char}	Mem.	T_{char}	Mem.
16	$1.08\mu s$	$16.25kbit$	$51.4\mu s$	$810.25kbit$
256	$17.24\mu s$	$260kbit$	$0.822ms$	$12.66Mbit$
PLL				
N_f	N_{cells}			
	1024		51840	
	T_{char}	Mem.	T_{char}	Mem.
16	$1.6ms$	$16kbit$	$3.2ms$	$810kbit$
256	$26.1ms$	$256kbit$	$50.9ms$	$12.66Mbit$

Table I
THE CHARACTERIZATION TIME AND MEMORY SIZE.

using different the clock synthesis methods and system architectures. Next, by analyzing the timing failure probability from a cell, we introduced a compact parametric model for it, and then sparsely sampled it to estimate the timing parameters. The third method considered the spatial correlation of the timings across the FPGA array to recover all the delays from a subset of measurements with a high accuracy. Evaluation results after implementing the methods on Xilinx Virtex 5 FPGA show the effectiveness of the proposed techniques in characterizing the FPGA cell delays with three orders of magnitude improvement in the characterization time while maintaining high accuracy.

REFERENCES

[1] J. S. J. Wong, P. Sedcole, and P. Y. K. Cheung, "Self-measurement of combinatorial circuit delays in FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 2, no. 2, pp. 1–22, 2009.

[2] P. Sedcole, J. S. Wong, and P. Y. K. Cheung, "Modelling and compensation for clock skew variability in FPGAs," in *International Conference on Field-Programmable Technology*, 2008, pp. 217–224.

[3] P. Sedcole, J. Wong, and P. Cheung, "Characterisation of FPGA clock variability," in *Annual Symposium on VLSI*, 2008, pp. 322–328.

[4] P. Sedcole, S. P. Wong, and P. Y. K. Cheung, "Compensating for variability in FPGAs by re-mapping and re-placement," in *International Conference on Field Programmable Logic and Applications*, 2009, pp. 613–616.

[5] M. Tahoori and S. Mitra, "Application-dependent delay testing of FPGAs," *IEEE Transaction on CAD*, vol. 26, no. 3, pp. 553–563, 2007.

[6] X. Li, F. Wang, T. La, and Z. Ling, "FPGA as process monitor-an effective method to characterize poly gate CD variation and its impact on product performance and yield," *IEEE Transaction on Semiconductor Manufacturing*, vol. 17, no. 3, pp. 267–272, 2004.

[7] E. Chmelar, "FPGA interconnect delay fault testing," in *International Test Conference*, 2003, p. 1239.

[8] M. Abramovici and C. E. Stroud, "Bist-based delay-fault testing in FPGAS," *Journal of Electronic Test*, vol. 19, no. 5, pp. 549–558, 2003.

[9] L. Cheng, J. Xiong, L. He, and M. Hutton, "Fpga performance optimization via chipwise placement considering process variations," 2006, pp. 1–6.

[10] H. Onodera, "Toward variability-aware design," in *Symposium on VLSI Technology*, 2007, pp. 92–93.

[11] P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nm FPGAs and beyond," in *International Conference on Field-Programmable Technology*, 2006, pp. 97–104.

[12] J. S. Wong, P. Y. K. Cheung, and N. P. Sedcole, "Combating process variation on FPGAs with a precise at-speed delay measurement method," in *International Conference on Field Programmable Logic and Applications*, 2008, pp. 703–704.

[13] J. S. J. Wong, P. Sedcole, and P. Y. K. Cheung, "A transition probability based delay measurement method for arbitrary circuits on FPGAs," in *International Conference on Field-Programmable Technology*, 2008.

[14] M. Brown, C. Bazeghi, M. Guthaus, and J. Renau, "Measuring and modeling variabilityusing low-cost FPGAs," in *International Symposium on Field-Programmable Gate Arrays*, 2009, pp. 286–286.

[15] M. Orshansky, S. R. Nassif, and D. Boning, *Design for Manufacturability and Statistical Design: A Constructive Approach*. Springer, 2007.

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

[17] F. Liu, "A general framework for spatial correlation modeling in VLSI design," in *DAC*, 2007, pp. 817–822.

[18] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transaction on Information Theory*, 2006.

[19] C. Hegde, M. B. Wakin, and R. G. Baraniuk, "Random projections for manifold learning - proofs and analysis," in *Neural Information Processing Systems*, 2007.

[20] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, no. 9, pp. 717–772, 2008.

[21] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 2, no. 1, pp. 1–33, 2009.

[22] "Virtex-5 FPGA data sheet, product specification," Xilinx Inc., Tech. Rep., 2009.

[23] M. Grant and S. Boyd., *CVX: Matlab software for disciplined convex programming*, 2009. [Online]. Available: http://stanford.edu/ boyd/cvx