Title

Analytical State Depended Timing Model for Voltage Scaled Circuits &amp; Low Overhead Delay-Based Physically Unclonable Function

Author

Pirbadian, Aras

Publication Date

2016

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
IRVINE

Analytical State Depended Timing Model for Voltage Scaled Circuits
&
Low Overhead Delay-Based Physically Unclonable Function

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

In Electrical Engineering and Computer Science

By

Aras Pirbadian

Dissertation Committee:
Associate Professor: Ahmed M. Eltawil, Chair
Professor Fadi J. Kurdahi
Associate Professor Rainer Doemer

2016

Dedicated to:

My beloved wife, Sara

And to my beloved parents Parisima and Ali

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

Firs and for most, I would like to express my appreciation and praise to my PhD advisor professor Ahmed Eltawil for his continuous patience, understanding and support during my graduate studies. My sincere thanks must also go to my co-advisor professor Fadi Kurdahi for his invaluable guidance and thoughtfulness during those years.

I am also most grateful to my best friend and wife Sara, for her encouragement, dedication and love during my PhD years. Her tremendous support, confidence and inspiration throughout my PhD was in the end what made this dissertation possible.

Also, my beloved parents, Parisima and Ali receive my deepest gratitude for their many years of unconditional trust, encouragement, endless patience, sacrifice and love. There is no way to express how much their support meant to me.

Finally, I can never forget my parents-in-law Roshanak and Ali, who I am always grateful for their generous inspiration and their firm believe in me.

CURRICULUM VITAE

2007	B.S in Electrical and Computer Engineering, Isfahan University of Technology
2009	M.S in Electrical and Computer Engineering, California State University Los Angeles
2016	Ph.D. in Electrical Engineering and Computer Science, University of California Irvine

FIELD OF STUDY

Wireless Systems and Circuits

PUBLICATIONS

A. Pirbadian, M. S. Khairy, A. M. Eltawil and F. J. Kurdahi, "State dependent statistical timing model for voltage scaled circuits," *2014 IEEE International Symposium on Circuits and Systems (ISCAS),* Melbourne VIC, 2014, pp. 1432-1435.
doi: 10.1109/ISCAS.2014.6865414

ABSTRACT OF THE DISSERTATION

Analytical State Depended Timing Model for Voltage Scaled Circuits
&
Low Overhead Delay-Based Physically Unclonable Function

By

Aras Pirbadian

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Irvine, 2016

Associate Professor Ahmed Eltawil, Chair

With the continued scaling of chip manufacturing technologies, the significance of process variation in performance of the systems is increasing. Specifically, process variation results in growing voltage and frequency overhead margins required to ensure error free operation of circuits. However, the traditional practice of overdesigning the systems to cover process variation is no longer an efficient design methodology in an age with high demands for processing power and limited energy supplies. In this dissertation, a novel analytical model is proposed to predict the required margin accurately in the early stages of design space exploration. The model can be used to optimize the system overhead in error free calculations or to release the bound by full correctness in error tolerant parts of systems and optimize the energy vs. performance trade-off. Additionally, this model also considers the statistics of the inputs of the circuit as compared to other existing efforts enabling it to achieve close predictions of full circuit simulation results in a short time. This model is finally used in an adaptive carry select/ripple carry adder configuration to demonstrate the potential achievable power savings.

Growing variation in newer technology nodes is not always a negative side effect. The increased inherent randomness in the process manufacturing technology can be utilized to develop unique physically unclonable functions (PUFs). These functions are irreproducible hardware-based authenticating systems, which do not require memory-based storage. A low overhead delay-based PUF using the variation of the silicon manufacturing is also proposed in the second part of this work. The proposed PUF uses a simple and efficient structure to convert the randomness of the manufacturing process into random responses to fixed challenges in identically designed circuits.

PART I: Timing Model for Voltage Scaled Circuits

CHAPTER I: OVERVIEW

I. Introduction

With the shrinkage of the geometries in the newer silicon manufacturing technologies, the effects of process variation gain prominence in affecting the system and circuit's performance and reliability. To overcome the challenge of making reliable systems in the smaller technology nodes, overdesigned and conservative systems have been used, which despite being reliable are not very power efficient. These designs, in general, add redundancy to the systems in a variety of forms to ensure correctness in all the calculating and storing components. However, there are many applications that by nature or by design tolerate some degree of error. As an example, wireless systems face the errors caused by the effects of their communicating channel. They are also designed to recover from such errors using a forward error correction (FEC) component. Even human eyes are largely insensitive to the pixel size color variations in an image when the frequency of such variation is small. Consequently, a system transmitting data over the air or an image compression or decompression algorithm does not necessarily need to be an error-free system as long as the inaccuracies do not exceed acceptable limits. In this work, a fast mathematical methodology has been proposed to quickly model the timing alterations resulted from the variations on the underlying circuitry of a system. Using this model, it is possible to predict the potential timing violation errors and to adjust the resources available to a system to achieve a more efficient system while not exceeding the desired acceptable error level.

II. Manufacturing Process Variation

The attributes of transistors fabricated on a chip are typically different from designed transistors due to the variations in their manufacturing process. The transistors’ performance is also a function of the physical parameters of the fabricated transistors as well as their operating environment. Recently, there is a great amount of attention and concern about the effects of variations of transistors on the yield and performance of the newer technology chips. These effects are so drastic that they have practically limited the development speed of the new technologies and have been indicated as major challenges for the semiconductor industry [1].

The environment of transistors can also have unexpected changes and its variations are categorized into supply voltage and temperature variation. The main source of supply voltage variation is the voltage drop across the resistance of the interconnections. Typically, many consumer units across a chip share the same supply voltage and its paths. These connections can have physical alterations due to process variation and sharing them on narrower metal layers in newer technologies worsens the supply voltage fluctuations. This is because the variation in the width of the connection is a larger percent of its total width. Additionally, the supply voltage variation directly affects the propagation delays of transistors, which will be discussed further in the next section. Alternatively, the main source of temperature variation is the changes in the neighboring transistors’ switching frequency and it will affect the transistor threshold voltages.

Transistors physical parameter alteration, which is a direct result of process variation on each transistor, can also be divided into systematic and non-systematic. Systematic variation signifies the deterministic portion of the process variation. This component of the variation is predicted from the relative distance of the devices as well as the location on the wafer or the patterns of the surrounding areas. This variation, which is caused by chemical mechanical

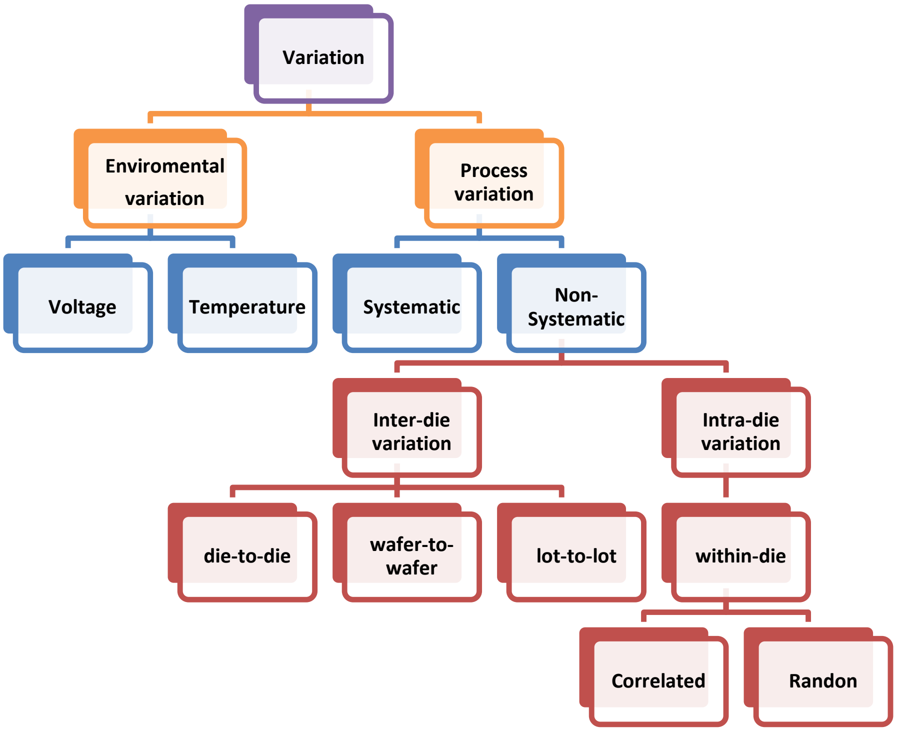


Figure 1. Classification of variation parameters [4].

polishing and proximity-based lithography, can be resolved by accurate modeling of the process [2] [3].

The non-systematic variation represents the unpredicted portion of the process variation and has a stochastic behavior. Smaller transistor gate sizes and increases in the number of transistors per unit area reduce the chances of manufacturing matching transistors. The limitations of the manufacturing equipment as well as other interfering mechanisms introduce these unfavorable stochastic variations into the process.

Figure 1 illustrates different categories of variation [4].

Non-systematic variation is also characterized depending on whether it is between multiple copies of the same transistor or if it is on two identically designed transistors on the same die. The first type, which is named inter-die variation, can be observed on transistors on different lots, from a wafer to the next or on multiple copies of the die on the same wafer. The key cause of inter-die

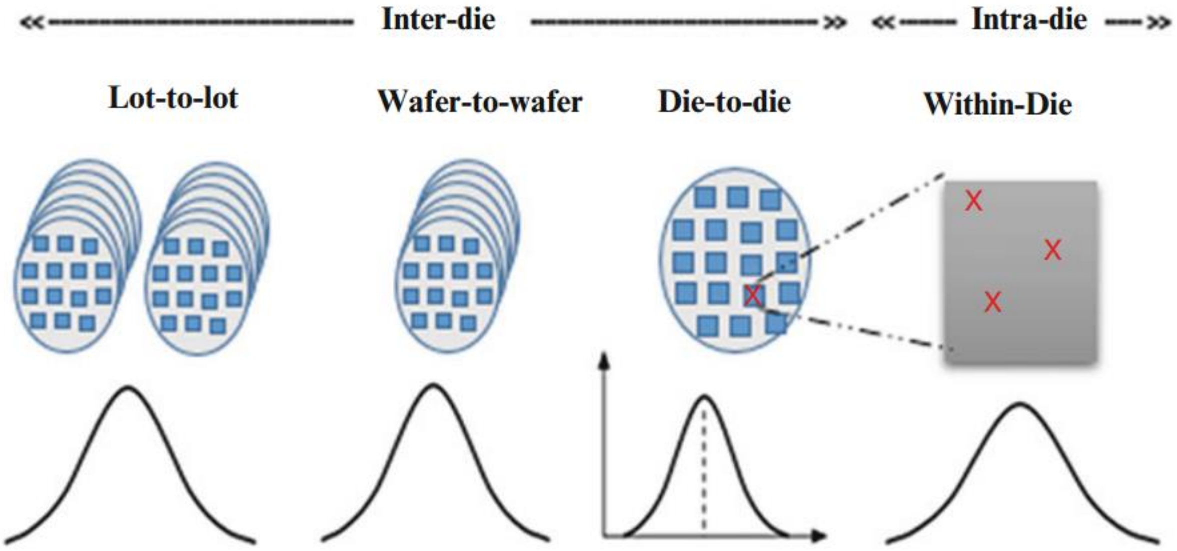
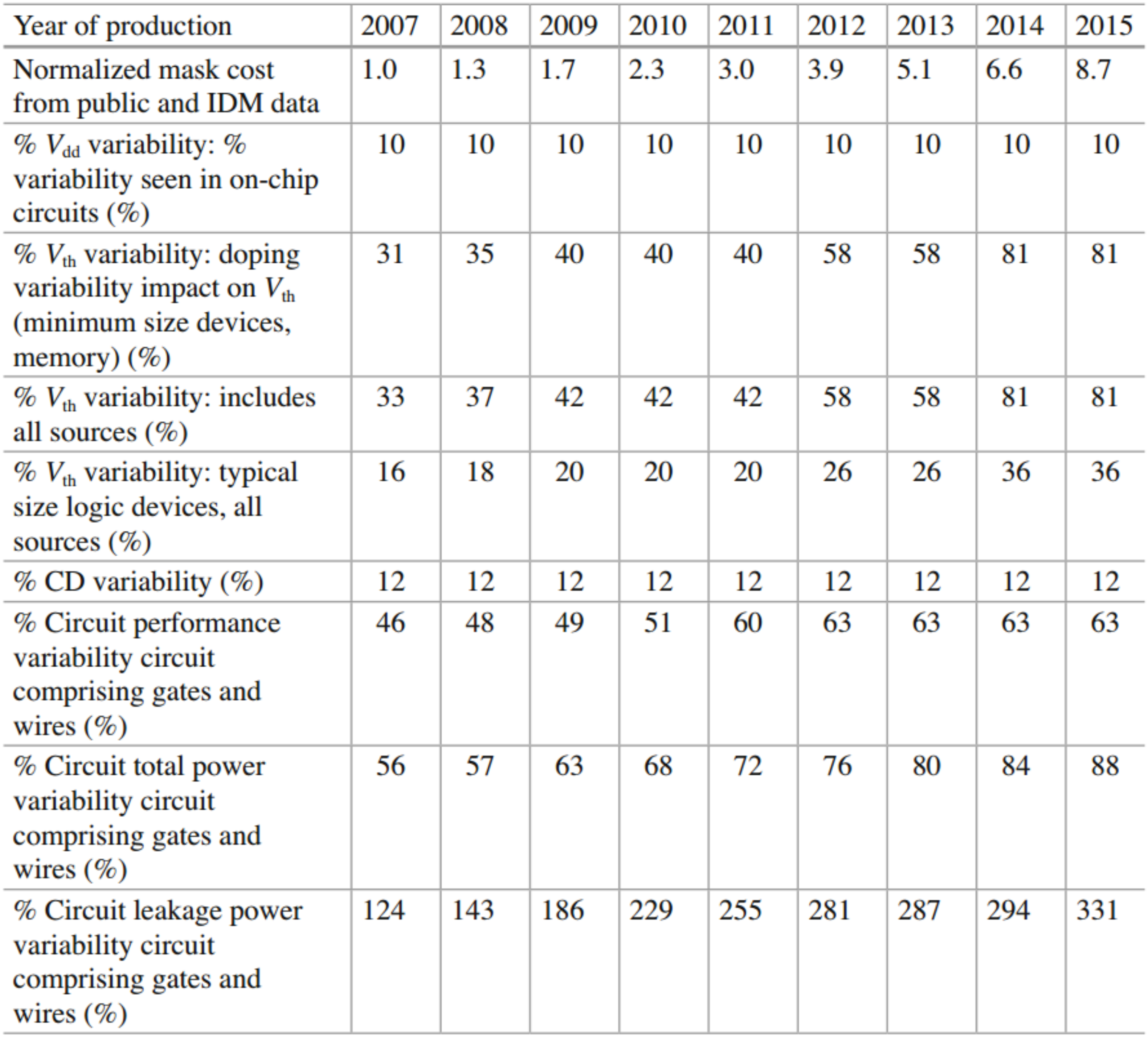


Figure 2. Inter-die and intra-die process variation [4].

variation is the technological restrictions, which result in the absence of sufficient control on the manufacturing process. The second type of non-systematic variation, which is named intra-die variation, diversifies the performance of identically designed transistors on a single chip. Despite being non-systematic, intra die variation can be correlated or random. The spatial correlation would be observed when closely spaced transistors show more similar variations than those placed far away [5] [6]. Sources of this variation such as etching, layout information, and lithographic effects can be modeled to reduce the effects of these correlations. Alternatively, the random portion of the intra-die variation does not have any meaningful spatial behavior. This variation results from random dopant variation and gate line edge roughness and is inevitable with the current manufacturing technology.

The process variation effects on transistors are observed on the physical parameters such as gate length, width, oxide thicknesses, threshold voltage, and numerous other physical parameters of a transistor [7] [8] [9]. The process variations effects can also be observed on the chip connections. Capacitance, resistance, and inductances of the interconnections in a die are also changed due to the process variation causing differences in the propagation delays of the signals. Table 1 summarizes the variations of different technology nodes as it was expected by the International Technology Roadmap for Semiconductors [1].

Table 1. Predictions of the international Roadmap for Semiconductors [1].



Year of production	2007	2008	2009	2010	2011	2012	2013	2014	2015
Normalized mask cost from public and IDM data	1.0	1.3	1.7	2.3	3.0	3.9	5.1	6.6	8.7
% V_{dd} variability: % variability seen in on-chip circuits (%)	10	10	10	10	10	10	10	10	10
% V_{th} variability: doping variability impact on V_{th} (minimum size devices, memory) (%)	31	35	40	40	40	58	58	81	81
% V_{th} variability: includes all sources (%)	33	37	42	42	42	58	58	81	81
% V_{th} variability: typical size logic devices, all sources (%)	16	18	20	20	20	26	26	36	36
% CD variability (%)	12	12	12	12	12	12	12	12	12
% Circuit performance variability circuit comprising gates and wires (%)	46	48	49	51	60	63	63	63	63
% Circuit total power variability circuit comprising gates and wires (%)	56	57	63	68	72	76	80	84	88
% Circuit leakage power variability circuit comprising gates and wires (%)	124	143	186	229	255	281	287	294	331

These variabilities in the device level cause some unwanted effects that require significant attention in the design process. The more random the physical variation in the device parameters, the more care will need to be given to the design. One of the key parameters of the transistor that is affected by the process variation is the threshold voltage. Transistor threshold voltage has been especially sensitive to the technology scaling because the random dopant fluctuations constitute the largest contributor of its variation and the number of dopants on newer technologies has dropped to tens of dopants. This has made transistors sensitive to the distribution and density of dopants and further spreads their performance. A change in the threshold voltage of a transistor will affect both the on and off currents. Transistors with different threshold voltages will turn on with varied currents at a fixed voltage. Since the leakage current also exponentially diverges by a varied threshold voltage, the ratio of the on to off currents of the transistors is also severely affected. A chip component that has transistors with diversified threshold voltages is also more likely to be extra sensitive to the supply voltage variations. In this case, variations in the supply voltage will cause the transistors to turn on and off with further diversified currents. The combination of these effects increases the probability of having transistors with unusual performances. Also, on a chip with a smaller technology node and multi-million transistors, a significant drop in the yield would be observed.

The most popular and the simplest form of handling the process variation is avoiding it by the use of excessive redundancy in various forms such as higher input voltage or slower operating frequencies. This method will ensure the correct performance of the circuit even in the face of variations in the transistor parameters and environment. However, it is becoming more difficult with the new technologies to disregard variation effects, and it also means the acceptance of lower yields. Therefore, such techniques would be wasteful in using resources.

Various other techniques have been proposed in the literature to model or analyze process variation and avoid low yield and performance. Specifically, three classes of techniques can be identified depending on the stage at which they have focused their effort. The first group is the set of techniques that attempts to model the statistical distributions of the circuit performance parameters and design the circuit to meet the required constraint. Techniques such as statistical static timing analysis (SSTA) and on-chip variation (OCV) fit into this category [10] [11]. These techniques will be further discussed in greater detail in the later sections. The second category consists of the techniques that detect the performance of the circuit after manufacturing process and adopt the frequency, supply voltage, or clock skew accordingly [12][13] [14]. Finally, the last category is techniques that avoid variation by synthesizing the circuits by characterizing the delay failures caused due to variation and dynamically changing the operation condition [15]. This technique can be further extended to the techniques that relax the system-wide correctness of the calculations [16][17] [18] [19] [20] [21] [22]. Such techniques argue that there is not always a need for correctness in all the calculations. There are parts of systems that are not very sensitive to small or infrequent inaccuracies, or they can be corrected later. One of the simplest of these techniques utilizes an error tracking mechanism that measures the rate of error on a voltage scaled circuit and redoes the operation when it exceeds the limits [23].

In this work, a combination of the first and last techniques are used to model the performance of an accuracy-relaxed system in a voltage- or frequency-scaled environment in the design phase. Such a model can be utilized to design a highly efficient system that is very conservative in using the resources and, therefore, power-performance efficient. The next section will discuss the voltage and frequency scaling of systems and their effects.

III. Voltage and Frequency Scaling

With exponentially growing appeal for processing power, increased number of portable devices, and crucial limitation of energy supply in battery operated devices, significant challenges are faced to meet the tight power per operation budget. Furthermore, the small size of many embedded systems limits the amount of acceptable heat dissipation. An increase of 15 degrees

Celsius in the temperature doubles the device failure rate by up to a factor of two [24]. This rate is even more vital in embedded devices used for medical purposes. In addition, it is estimated that information and communication technology (ICT) contributes 3% of the world's overall carbon footprint and any reduction in this figure would help to have a greener environment [25]. Also, the creation of devices with lower power consumption allows the use of smaller battery sizes, reducing the weight, area, and cost. To meet these demands, an efficient power management technique is required [26]. One solution, dynamic voltage and frequency scaling (DVFS), was initially introduced by Wiser et al. [27]. In their paper, Wiser et al., introduce a metric to measure the performance of a CPU per energy unit consumed and suggest a dynamic adjustment of the voltage and frequency. Using this metric, they show that it is possible to save up to 50% of power consumption under conservative circuit design assumptions. These are savings on top of the obvious methods of stopping the CPU in the idle loops.

The following equations show the relationship between the power consumption, supply voltage, and the frequency of operation on a transistor

$$P_{Total} = C_{avg}V_{DD}^{2}f + P_{Static} \qquad [1]$$

$$P_{Static} = I_{leak}V_{DD} \qquad [2]$$

where $C_{avg}$is the summation of the gate, diffusion, and wire capacitances and I_{leak} is the leakage current [26] [28]. This equation indicates that a reduction in the supply voltage or the frequency of operation can reduce power consumption. However, an increase in the frequency of the operation of a circuit will result in possible timing failures on paths that have propagation delays longer than the new clock period. Furthermore, a reduction in the supply voltage will extend the propagation delays of logic gates and it can also result in timing violations. The following equation shows the relationship between the propagation delay time and the voltage scaling [29].

$$t_{pd} = \frac{C}{(v - v_t)^{\alpha}} \qquad [3]$$

Here, C and α are constants found by curve fitting the change of the propagation delay relative to the scaled voltage and v_t is a technology dependent constant. These equations indicate that any adjustment on the voltage or frequency of operation must be with full awareness of changes on the propagation delay and possible timing failures.

With the power consumption quadratically dependent on the supply voltage, dynamic voltage scaling (DVS) has emerged as a simpler technique used to solely adjust the supply voltage according to the necessary throughput. The basic idea behind this method is that not all parts of a system are always fully utilized. Additionally, each particular chip, depending on its wafer corner, has specific capabilities. Conventional DVS takes these differences into consideration and by maintaining a safety margin to ensure full correctness, scales the voltage of less demanded components of a system to conserve power. This margin is required to cover for changes resulting from variation and to ensure timing failures do not occur. However, with the scaling of geometries in the new technologies and higher variations on the transistors and interconnecting paths, the required margin is growing. This increased margin translates into larger voltage overhead and maintaining this safe boundary proves to be energy inefficient. A more robust approach named variation-aware DVS first proposed by [4] suggested to eliminate this bound by correctness. It is shown then that by monitoring the rate of inaccuracies caused by a dynamically-scaled voltage a system can achieve an optimized power consumption. This technique usually consists of performance manager software, which predicts the future performance needs of the system and adjusts the supply voltages accordingly. In this method, by having multiple levels of supply voltage, for sensitive and non-sensitive operations, the power consumption is controlled dynamically.

A similar method can also be pursued solely by adjusting the frequency of operation or dynamic frequency scaling (DFS) [30]. When parts of the system are not fully utilized, reducing the rate of charge or discharge of the capacitors on the system or, in other words, its clock frequency reduces the power consumption. As an example, when the rate of access to a memory unit drops, a reduction in the clocking frequency can provide a power saving while causing no interruption to the normal operation of the system. Similar to DVS, the requirement of full correctness on all operations can be dropped in a DFS, in this case, causing the propagation delay of non-synchronized operations to be possibly longer than the scaled clock period. This will result in inaccuracies; their effects need to be considered using error-aware techniques.

The three process variation handling techniques discussed in the last section can utilize the DVFS to achieve efficient systems in smaller technology nodes. Among the three methods, modeling the performance of signals under variation effects allows the designer to have a better understanding of the circuit behavior at the design phase. Timing analysis (TA) and on-chip variation (OCV) techniques are the most widely used techniques to model the circuit behavior. The next two sections discuss different flavors of these techniques in greater detail.

IV. Timing Analysis

Every fabricated sequential integrated circuit has a highest clock frequency under which it can operate correctly in a given environment. As the circuits become more complicated, finding this highest operating frequency at the design phase is becoming a more challenging task. Timing analysis techniques are simulation-based approximation methods to determine the expected delay time for the output of a combinational circuit to stabilize without the need for a full circuit analysis or tracing of all paths. In other words, they are design automation algorithms that offer a faster alternative solution for timing problems in logic circuits, which can save significant redesign and rebuild time. These algorithms, which were first developed as part of the design verification of IBM 3081, find out if all the design paths meet the required timing criteria [31]. That is that the data signals arrive at synchronized components early enough to satisfy the required setup time but not too early to interrupt the previous clock cycle of the next component. As an output of the algorithm, timing analysis produces a slack time at each block, which is the difference between the latest time a signal can arrive at a certain node and the actual latest arrival time of to that node. This slack time can then be used to determine the highest possible operating frequency with the presence of variation or to change the design where needed to increase the highest clock frequency. Timing analysis can be characterized into two categories of static timing analysis (STA) and dynamic timing analysis (DTA). STA is also further extended into two categories of deterministic static timing analysis (DSTA) and statistical static timing analysis (SSTA) [32] [33]. The next sections will discuss these techniques in more details.

A. Static Timing Analysis (STA)

Static timing analysis (STA) or deterministic STA is one of the simplest and widely adopted timing analysis algorithms, which ensures the correct operation of the fabricated chip. This algorithm validates all possible paths of a digital circuit for a timing violation. STA simplifies the calculations by the assumption of the fixed process parameters such as temperature, voltage, gate length, oxide thickness, and etc. which allows the analysis of circuits with multi-million transistors. It also overestimates the delay of the longest path and underestimates the delay of the shortest path, which guarantees that the design will operate at least as fast as predicted and effectively addresses false path and multiple cycle path issues. Combined with the simplicity of STA and its linear increase of the runtime with the increase of the circuit size, these factors are the main reasons for its popularity [34]. However, at the nanometer scales, process variation causes the underlying deterministic assumption of the STA to be challenged while STA is unable to incorporate the process variations. In addition to this, for every process parameter the STA algorithm needs to run separately. Therefore, as the number of variability sources increases, this corner-based technique is becoming computationally expensive. Furthermore, STA generates no information to simplify the given design specifications [32]. To address these issues with the deterministic STA the statistical STA (SSTA) has been proposed. The next section will review the SSTA in greater detail.

B. Statistical Static Timing Analysis (SSTA)

STA was an effective tool to analyze the timing of the circuits and to find the highest frequency of operation for many years. However, with the increased variation in the gate delays due to the process voltage and temperature (PVT) variations, the deterministic nature of the STA is no longer a viable assumption. To address this matter, statistical STA (SSTA) has been recently proposed [10] [35] [32]. The strength of SSTA, which is divided into two categories of parametric and Monte Carlo methods, is that it models the process variation parameters as random variables with known distribution. Unlike STA, which propagates the deterministic gate values and finds a

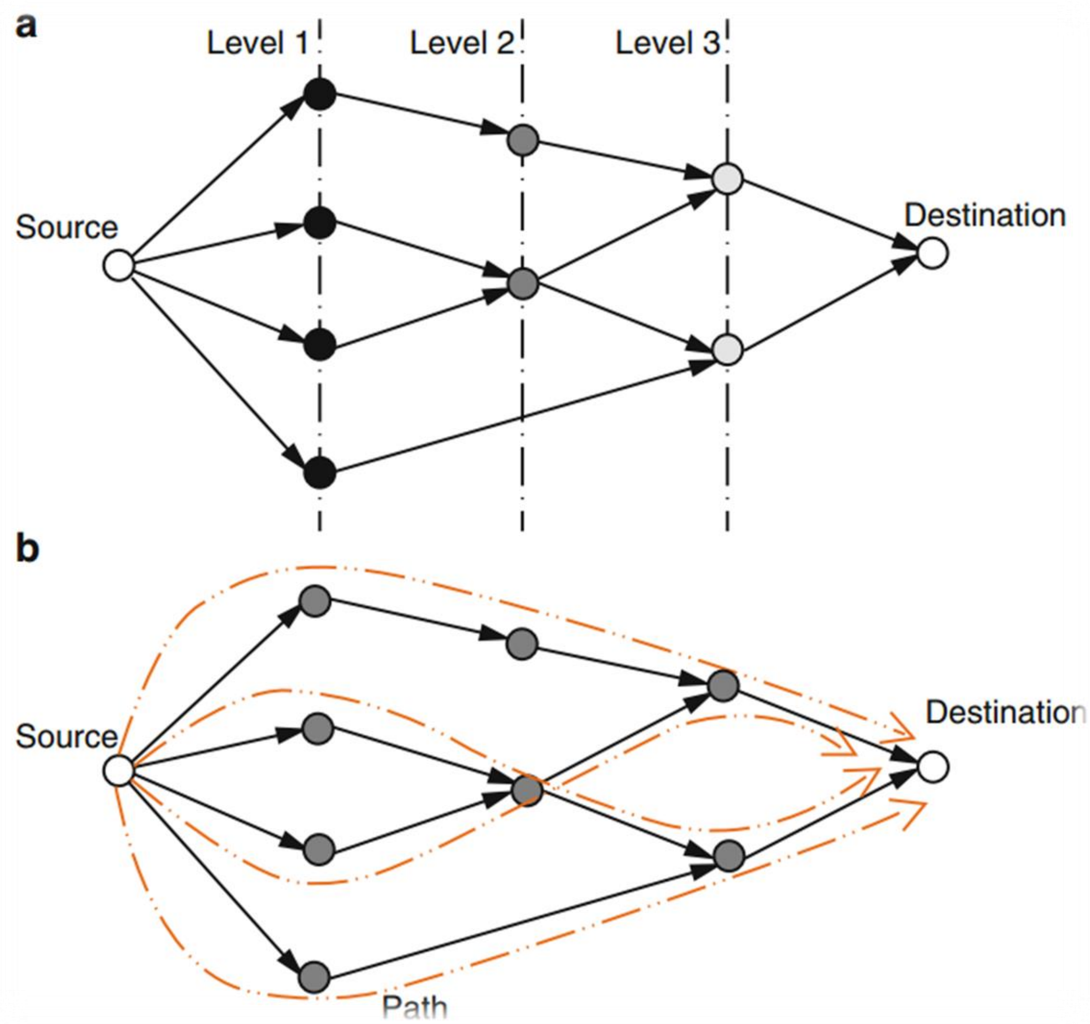


Figure 3 (a) Block based vs (b) Path Based SSTA [4]

one value delay and a fixed slack, parametric SSTA propagates the distribution of the falling and rising edges of the delays, therefore, arriving at a distribution for the last falling or rising edge of a signal and a distribution for the slack. Parametric SSTA can be divided into two categories, block-based SSTA and path based SSTA. Figure 3 shows these two approaches. As shown in Figure 3**Error! Reference source not found.** (a) in the block-based SSTA the circuit is analyzed in a hierarchical fashion. In this approach, each interconnection is taken as blocks. At each block, a statistical max operation is applied between all of the paths arriving. When all of the branches and nodes have been considered, a distribution for the max delay is achieved. The runtime in this method is linear, which makes it a fast operation capable of running on larger circuits. The other technique in the parametric SSTA is the path-based approach. In this method, a set of possibly critical paths is identified. A statistical analysis of the delay of each path is then performed by convolving the delay distributions of all the edges of each path and then taking the statistical max of all paths at the sink node. The challenge of this method is to find all the paths with some chance of being a critical path, which could result in an exponential increase in the runtime, especially in the more balanced circuits. The two approaches in the parametric SSTA differ in the precision and the computational cost. The block-based approach is a faster method while the path-based approach is more accurate [34]. It is important to note here that even though parametric SSTA is capable of finding the statistical distribution of the last falling or rising edge of the most critical path, it is not designed to report any timing distribution before the last edge. This is due to the use of the statistical max operation at each node, which only takes the distribution of the last change in the signal [10]. However, voltage and frequency over scaled circuits require knowing the signal distribution before the last edge more accurately to determine the optimum voltage or frequency operating point, which is the point that the circuit uses the minimum power with the least tolerable error rate.

Circuits that do not have an analytical description such as simulations of circuits with the existence of process variation can be processed using the Monte Carlo method. In this technique, using the probability distribution of the process variation, the samples of gate delay distribution parameters are drawn. These parameters are then used to run a simulation iteration of the Monte Carlo process for the specific circuit modeled to obtain the distribution of the delay of the circuit outputs. This method is very accurate since it is based on the actual simulation of the circuit. However, since many iterations of the simulation are required before the distribution is known, it is a very slow process. Therefore, its application is limited to small and medium size circuit [36] [4].

C. Dynamic Timing Analysis (DTA)

Static timing analysis (STA) checks static delays of a circuit's signals without considering the input or output vectors. In other words, STA ignores the dependence of the circuit timing to the change in the values. However, in practice, the gate delays are dependent on the input and output values or states of the circuit. Dynamic timing analysis (DTA) is a full simulation timing analysis method that increases the accuracy of the statistics by considering the effects of the input and output states in its analysis [4]. This will reduce the pessimism incorporated in the STA and is especially suitable for circuits that clock cross multiple domains. DTA uses full circuit simulation for different input vectors, which is a slow process and limits the application of DTA in many larger size circuits.

V. On-Chip Variation (OCV) Analysis

As discussed in the previous section, statistical STA attempts to find the statistical distribution of the signal’s arrival time by considering the effects of random process variations on each path and gate. Nevertheless, it ignores the deterministic variation in these components. As an example, designers typically analyze the circuit for different corners of the wafer without modeling variation effects that are observed on the same gates within a die. They have also historically used variation variables uniformly across the chip disregarding the physical information of the gate or path being modeled. However, integrators of larger and more complex chips have started examining the effect of deterministic variation on the overall performance. On larger size chips, the paths typically change layers more frequently. Since the layers are deposited independently, one layer might have a high delay characteristic while the next would be a fast one. Due to these effects, even under extreme cases, a long path would see more averaging effects than a best or worst case scenario might predict. This will make SSTA predictions more pessimistic than the real propagation delays. An alternative modeling method to the constant distributions used in STA quantifies the deterministic effects of the variation to change the distributions further across the die. The deterministic variation can be predicted from the location on the wafer or the pattern of the adjacent components and relate to the proximity effects, density effects, the relative distance of devices and gradients specific to each processing tool [37].

To model these non-random effects on-chip variations (OCV), techniques use derating factors in the static timing analysis to slow down or speed up certain gates or interconnects depending on their physical information. These derating factors are based on empirical measurements on silicon and indicate how variations are related to the device locations [38]. Different OCV techniques are under development to analyze the circuit behavior accurately.

Advanced OCV, which is the most established variation, will be discussed in the next section.

A. Advanced On-Chip Variation (AOCV)

Advanced OCV or location-based OCV is an easy to adopt solution to close the gap between the measured and predicted timing performance of the circuits by considering the location of the component and the surrounding patterns. The basic idea is to vary the distribution of the gates and nets in the analysis depending on the neighboring parts or the loading effects, which will lead to reduced timing slacks, less timing violation, and longer timing closure cycles. AOCV determines the best derating values by finding how much a specific path in a design is affected by the process variation [37]. As an example, in a normal timing analysis method, all buffers of a buffer chain with a nominal delay of 20ns will be treated equally, and a derating factor of one will be assigned resulting in a delay of 20ns xN at stage N. However, statistical HSPICE models and Monte Carlo analysis show that the deeper timing paths have less variation. AOCV assigns different derating factors to the stages of this chain buffer modifying the distributions of delays at each stage. Another effect measured by simulations of the variations in circuits is the relation between the variation and the number of transistors. It has been observed that cells with many transistors will exhibit less variation. Therefore, their derating is reduced in AOCV. In addition to the derating of the gates and nets timing distributions, it is possible to specify derate for guard bands to model non-process related effects such as voltage drops across paths. The total derating factor is then the multiplication of both derating factors [37] [39].

The next section will cover a more complex version of OCV, which allows additional timing margin by taking more factors into consideration and is especially needed for lower technology nodes.

B. Statistical On-Chip Variation (SOCV)

With further reduction in gate sizes the variation increases so much that unless an accurate timing characterization is implemented, the benefit of reducing the gate length can be questioned. Despite the good efforts of AOCV to predict the timing performance of the signals with derating factors, it only accounts for the variation purely based on the stage number and the type of the gate. In other words, it disregards the dependence of variation on slews and loading of the gates. Furthermore, gates on paths with relatively more complex components will likely behave differently than on a path with simple gates. Another improvement on SOCV is the variation on the slew, especially at lower voltages. Variations in slew rates have effects both on the delay through the cell as well as the output slew rate. Moreover, SOCV captures the setup and hold time variation, which changes the statistics of the available timing slack [40].

Overall, SOCV is capable of providing a better timing distribution by capturing more effects into account and accurately processing all statistical calculations in a reasonable time. However, it is a tool primarily designed for determining the best clock frequency at lower gate sizes. In addition, it is not the purpose of its creators to capture the timing distributions before the last event of the most critical paths. Therefore, it is not the preferred tool for a voltage or frequency scaled circuit, which intentionally pushes the circuit into regions with some inaccuracies.

CHAPTER II: ANALYTICAL STATE DEPENDENT TIMING MODEL

I. Introduction

Along with technological advances in shrinking silicon manufacturing dimensions, the variation of the physical parameters of the new technology chips have increased considerably. This effect is so substantial in smaller technology nodes that it has led to questioning the practicality of smaller gate sizes. In fact, the effect of variation in the performance, combined with the extra costs associated with manufacturing chips in higher technologies, has challenged further reduction of the gate sizes. Meanwhile, due to the high demand for devices consuming less energy, new techniques such as dynamic voltage and frequency scaling (DVFS) have gained significant momentum. Such techniques can be used to decrease the growing voltage or frequency margin needed to avoid variation effects and save energy. However, better use of such techniques requires accurate modeling of the distributions of the signals to optimize the energy savings. This work presents an energy reduction strategy, which rapidly and accurately models the full delay distribution of combinational circuit signals. This model can reduce the excessive margin used in the overdesigned circuits by accurately modeling the slack time between the signal and clock. However, it goes one step further compared to other similar modeling efforts in the literature with incorporating the effects of input transition into the analysis. It was shown previously in [42] that the propagation delay of gates vary according to the input state transition of logic gates. Incorporating these changes enables this analytical model to match the full circuit simulation delay distribution of signals in a small fraction of the time. This is because of the relative speed of analytically propagating the signals through the circuits compared to a Monte Carlo based simulation running the entire circuit for every sample picked. This increased speed allows the proposed model to be used for larger circuits. There are some similar SSTA work as in [43] considering input vector to exercise false paths and the Monte Carlo approaches running the entire circuit many times. However, to the best knowledge of the author, none of these techniques and no other similar work considered the input vector state dependency for deriving timing distribution with an analytical approach.

II. Methodology

There have been many previous works indicating that the distribution of propagation delays of gates under voltage scaling can be estimated accurately with a Gaussian distribution [44]. The mean and variance of this distribution depends on the previous and current states of the inputs to the gate and can be obtained at the characterization phase. Consequently, for a sample gate with n inputs, there would be 2^{2n} possible input transition sets, which correspond to 2^{2n} sets of mean and variance for the Gaussian distribution for every voltage. Furthermore, [42] proposed a method for obtaining the propagation delays of arithmetic circuits such as adders and multipliers under voltage scaling, which shows a very close match compared to HSPICE circuit simulation while achieving a speed gain by several orders of magnitude. However, this model employs dynamic timing analysis (DTA) combined with a Monte Carlo approach, which still requires an excessively long time to obtain the timing distribution for large circuits. As an example, for an 8-bit adder, an input vector set of size 4-gigabit input samples is required to cover all possible input combinations. This is a major concern for more complex circuits. In this work, an analytical approach is proposed to rapidly obtain accurate timing propagating delays of circuits under voltage scaling and achieving a speed gain of one to two orders of magnitude as compared to [42].

A. State Dependent Model

A state-dependent analytical, statistical model for the propagation delay of logic circuits is proposed here. The model uses the input transition states coupled with the logic function of the gate to obtain the timing distribution of the output transitions. For an output transition from i to j $(i \to j\,;\, i,j \in \{0,1\}), T_y^{i\to j}$ is defined as the timing distribution of output y for the transition from i to j. For a simple logic gate with two inputs and one output there are sixteen different input transitions (when considering current state and next state) and four possible transitions at the output. Of these four output transitions, two correspond to $1 \to 1$ or $0 \to 0$ changes and, therefore, result in zero propagation delay. This can be represented by a delta function such as

$$T_y^{0\to 0} = \delta(t)\,, T_y^{1\to 1} = \delta(t) \tag*{[4]}$$

The timing distribution of the other two transitions can be modeled as a sum of scaled Gaussian distributions

$$T_y^{0\to 1} = \sum_i \beta_i^{0\to 1} \times N(\mu_i\,, \sigma_i\,) \tag*{[5]}$$

$$T_y^{1\to 0} = \sum_i \beta_i^{1\to 0}{}_i \times N(\mu_i\,, \sigma_i\,) \tag*{[6]}$$

where $N\ (\ \mu_i\,, \sigma_i\ )$ is the normal distribution with mean μ and standard deviation σ of the *i*th input transition that will change the output from 0 to 1. Note that the proposed model can be extended to assume non-Gaussian input distributions. $\beta_i^{0\to 1}$ represents the probability of occurrence of the *i*th input change. Using these output timing distributions, the total propagation delay distribution is the weighted sum of all four transitions, which can be represented as:

$$T_y = \sum_j \sum_j \alpha_{i \to j} \times T_y^{i \to j} \qquad [7]$$

where $\alpha_{i \to j}$ is the weight of the output transition $i \to j$ which can be obtained via knowledge of the logic function of the gate, probability and number of input transitions that generate the $i \to j$ transition at the output.

B. Gate Look-Up Tables

The $N\ (\mu_i\,, \sigma_i)$ in the aforementioned equations is the Gaussian distribution of the propagation delay of the *ith* input transition. This distribution will need to be known prior to use of the above equation and is gate and input transition dependent. To obtain this distribution an HSPICE simulation is setup to simulate the effects of transistor variation on the gate propagation delay distribution. In this model all transistors are assigned a threshold voltage with normal distribution variation and standard variation according to the following equation:

$$\sigma_{vth} = \frac{C}{\sqrt{W * L}} \qquad [8]$$

and a mean that is equal to the threshold voltage in the absence of the variation [44]. L and W in the above equation are the length and width of the transistor respectively, and C is a technology-dependent constant. To obtain $N\ (\mu_i\,, \sigma_i)$ for all inputs a Monte Carlo simulation is set up to characterize the distribution of delays for all 2^{2n}possible input vectors with n being the number of inputs to the gate. This input vector space will consider all of the possible transitions from the previous state to the current state of the input. The simulation measures and stores the propagation delay of each run of the circuit with a given input. The mean and standard deviation of all runs with a given input transition are then extracted and used in the row number i of a look-up table

(LUT) with 2^{2n} rows. Since the propagation delay of gates are voltage dependent, there would be one LUT for each voltage of each gate covering all possible input transitions for that gate. The key in this simulation is that since the LUTs are made per gate, the number of inputs and transitions is significantly smaller than the number of inputs in a circuit. Therefore, this characterization would use significantly less time compared to a circuit-level Monte Carlo simulation. Moreover, the characterization would be performed only once per implementation of the gate and circuits can be modeled using the proposed model afterward.

C. Intermediate Transitions

Digital logic circuits are implemented as cascaded chains of individual logic gates. The previous section explained the statistical timing distribution of a standalone logic circuit. In this section, we consider cascaded logic blocks. As an example, consider the two cascaded logic gate blocks shown in Figure 4. To determine the timing distributions of the transitions of Y_2 and their

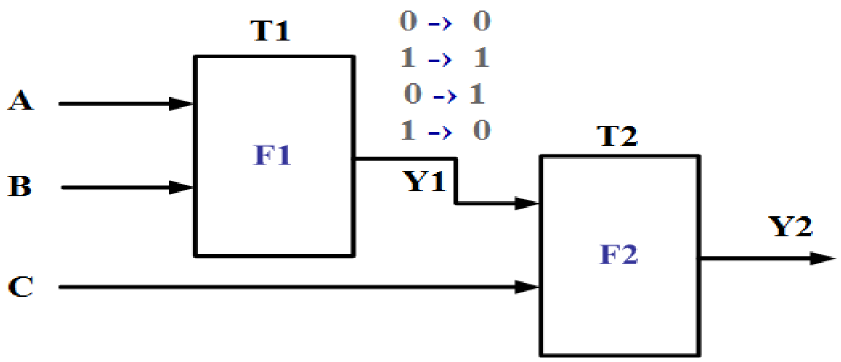


Figure 4. Cascaded logic gates

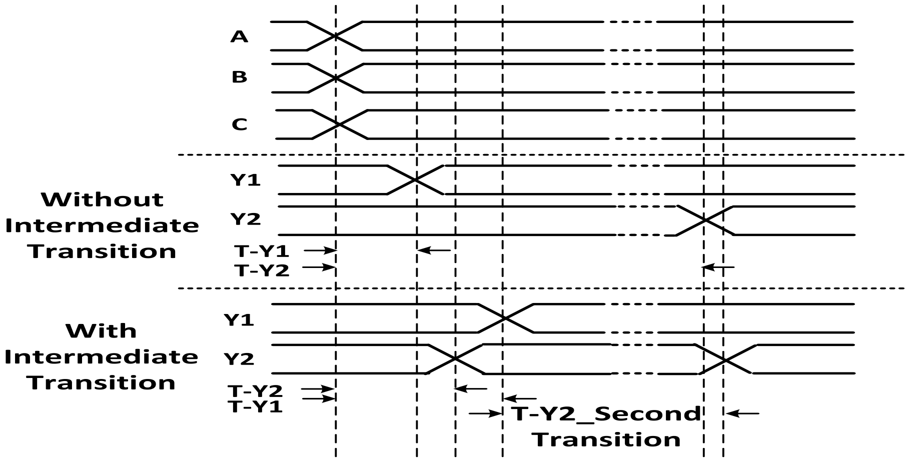


Figure 5. Timing diagram of logic nets

corresponding probabilities, the intermediate transition of Y_2 (or glitches) needs to be considered. Generally, there are two possible scenarios for the output Y_2 transition: 1) Y_2 does not experience an intermediate state and the change in Y_2 is solely dependent on Y_1 delay and F_2 gate delay, and

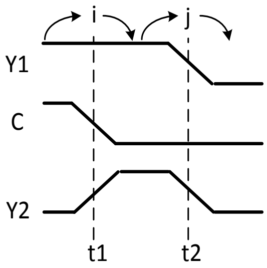


Figure 6. 11 to 00 transitions for the XOR circuit.

2) Y_2 experiences an intermediate state due to other input changes prior to settling to a final state due to a change in Y_1. In the first case, as explained by the cascaded circuit in Figure 5 and its timing diagram in Figure 5, the total delay timing of the output is the addition of the timing delay of both blocks. Equivalently, the timing distribution can be expressed as the convolution of delay timing distributions of both gates. An example showing the second case, where Y_2 experiences a glitch, is shown in Figure 6. The timing delay in this scenario is somehow complicated and will be explained further with an example. To better explain the timing model with intermediate states and without loss of generality, we consider an example of a two XOR circuit where the blocks of Figure 4 are replaced with XOR gates. Figure 6 shows a case where the inputs of the logic gate F_2 go through a [(Y_1C); 11→00] transition. In this figure, t_1 is the transition time of C and t_2 is the time of the Y_1 transition. The transition for Y_2 shown in the figure is hypothetical and assumes zero propagation delay in the F_2 gate. Although the output should not change for such a transition, an intermediate state will occur. To identify the timing distribution for all possible 1 → 0 transitions, including those contributed by glitches, one must account for the relative delays of the two gates. To do so, the timing distribution of F_1 is multiplied with a step function centered at the mean of the timing distribution of F_2. Thus the following equation is the contribution of the 11 → 00 transition at F_2's input to the output 1 → 0 transition.

$$N\left(\mu_j, \sigma_j\right) * U(\mu_\mathrm{i}) \times T_{y1}^{1\to 0} \qquad [9]$$

where $T_{y1}^{1\to 0}$ is the timing distribution of the output Y_1 transition from 1 → 0 as calculated by (3) and * is the convolution operation. $U(\mu_i)$ is a step function rising at the mean of the second gate's transition and $N\left(\mu_j, \sigma_j\right)$ is the F_2's Gaussian timing distribution for the given inputs. In the 11 →

00 transition shown in the figure the values i and j represent the indices of the gate F_2 input transitions of 11 → 10 and 10 → 00 respectively.

The total distribution for a given output transition is the addition of the timing distributions of both scenarios and can be expressed as:

$$T_{y2}^{1\to0} = \sum_{i,j} N\big(\mu_{\mathrm{j}}, \sigma_j\big) * \; U(\mu_{\mathrm{i}}) \; \times \; T_{y1}^{1\to0} \qquad [10]$$

$$+ \sum_{i} N\big(\mu_{\mathrm{j}}, \sigma_{\mathrm{j}}\big) * T_{y1}^{1\to0}$$

The previous equation can be simplified as expressed in (7).

$$T_{y2}^{1\to0} = \sum_{i,j} T_{y1}^{1\to0} * \big(1 + U(\mu_{\mathrm{i}})\big) \times N\big(\mu_{\mathrm{j}}, \sigma_{\mathrm{j}}\big) \qquad [11]$$

Finally, the total output distribution is calculated as in equation (4). Considering that all input values for the XOR circuit are equally likely, then all possible input transitions at F_2 will be equally likely and the weights would be equal to 1/16.

D. Error Injection Model

The main objective of this statistical model is to find the possible timing violations and reflect them as errors into the logic circuit model when it is voltage scaled. Figure 7 illustrates the equivalent model consisting of an error-free circuit combined with an error injector such that the final output statistics match those of the voltage-scaled circuit. The model derives the distribution of the voltage-scaled logic circuits based on the initial and final values of the input and the output bits and injects the error by flipping the output bit whenever the signal requires a longer time than that assigned.

Consider as an example a 4-bit adder; the delay distribution per each output bit transition is obtained based on the proposed analytical model. The probability of error per output word is then calculated by considering all of the possible bit combinations. The errors are then injected at the output of the adder based on the probability of each magnitude of error for the given initial and final values of the output. Figure 8 shows the probability of each magnitude of error based on our model for a 16 to 0 output transition for 100ps clock frequency.

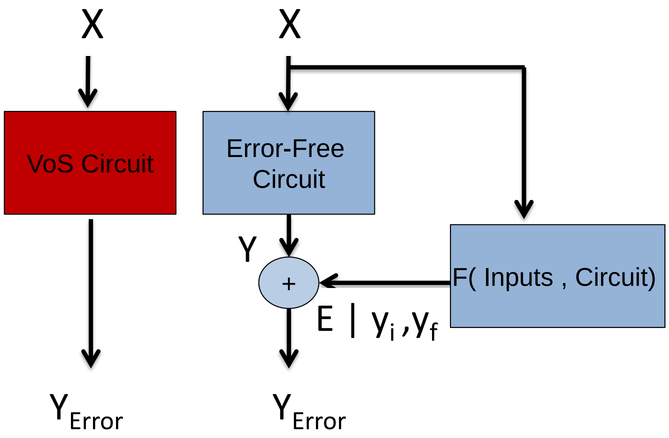


Figure 7. Equivalent model of erroneous voltage scaled circuit

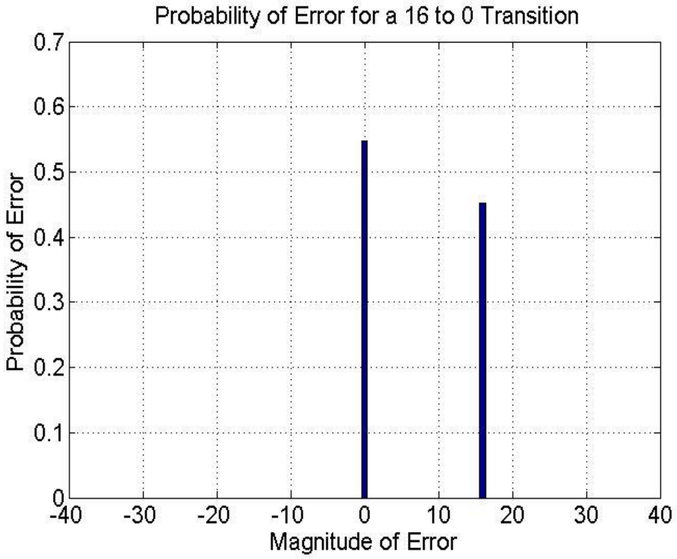
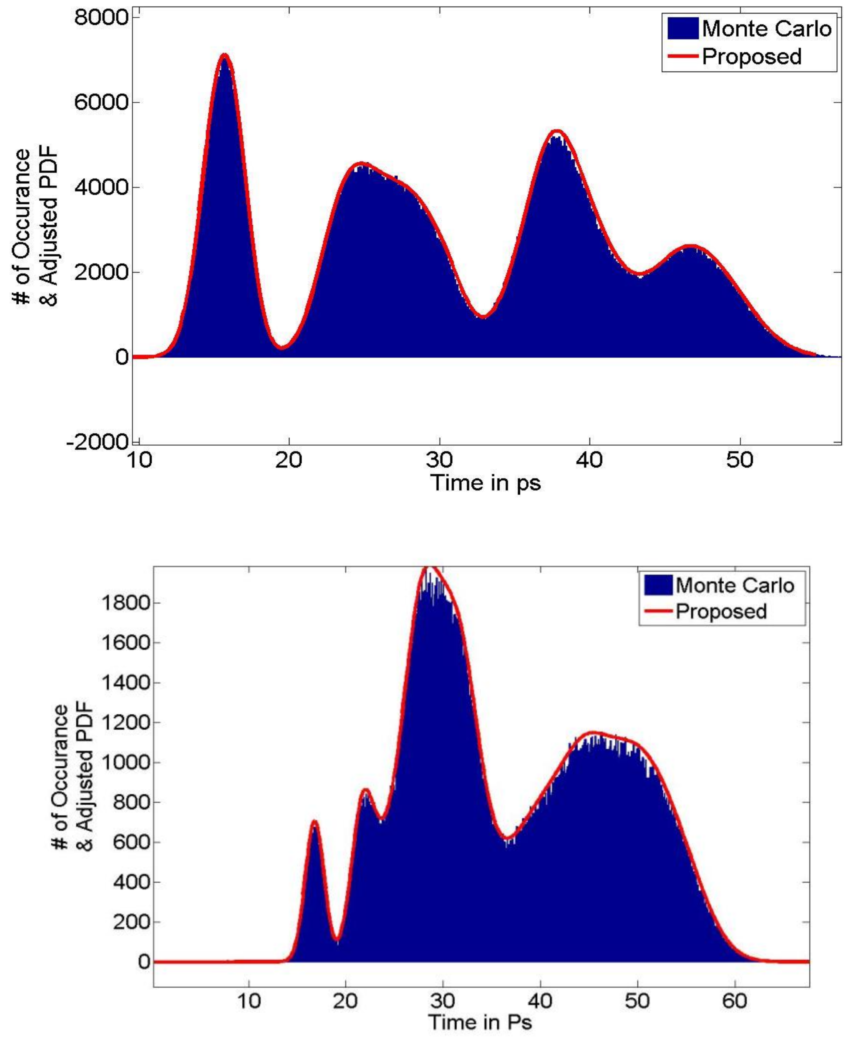


Figure 8. A sample of the error density function

III. Model Verification

To verify the accuracy of the proposed model, different logic circuits have been tested. We illustrate a comparison between the timing distributions obtained by the Monte Carlo method used in [42] and the proposed statistical model for (a) two cascaded XOR gates and (b) a two-bit adder.

Figure 9 (A) illustrates the close match between the timing distributions of propagation delays using the analytical method displayed as a solid line versus the Monte Carlo method, displayed as histogram bars for the XOR circuits. Figure 9 (B) provides the results for the last carry bit of a two-bit adder. Again, the analytical method accurately predicts the timing distribution at the output. One of the major advantages of the analytical method is the increase in processing



(A)

(B)

Figure 9. Hspice/Monte Carlo simulation results (bars) versus proposed statistical model (Solid lines) for (A) XOR circuit on top and (B) adder on bottom

speed as shown in Table 2, which illustrates a comparison of the processing time for the HSPICE simulation, Monte Carlo method suggested in [42], and our proposed method for various circuits.

IV. Case Study 2p-IDFT

To demonstrate the proposed model at the application level, we consider an image compression application (e.g. JPEG) illustrated in Figure 11 utilizing a 2-point discrete Fourier transform (DFT). The 2-point DFT is essentially an addition and subtraction of two inputs as shown in Figure 12. A sample image is quantized, and every 2 pixels are fed into an error-free 2-point DFT. The error probabilities are then calculated based on the initial and final values of the DFT output. Error magnitudes are chosen based on the error probabilities calculated by the proposed model. These errors are added to the output of each DFT output. The error-free inverse DFT and de-quantization are generated to reconstruct the sample image.

Table 2. Run time comparison

	XOR	Full Adder	8-bit ripple adder
Proposed Model	7 Sec	10 sec	25 sec
Monte Carlo**Error! Reference source not found.**	6 minutes	8 minutes	35 minutes
Spice	10 hours	53 hours	215 hours

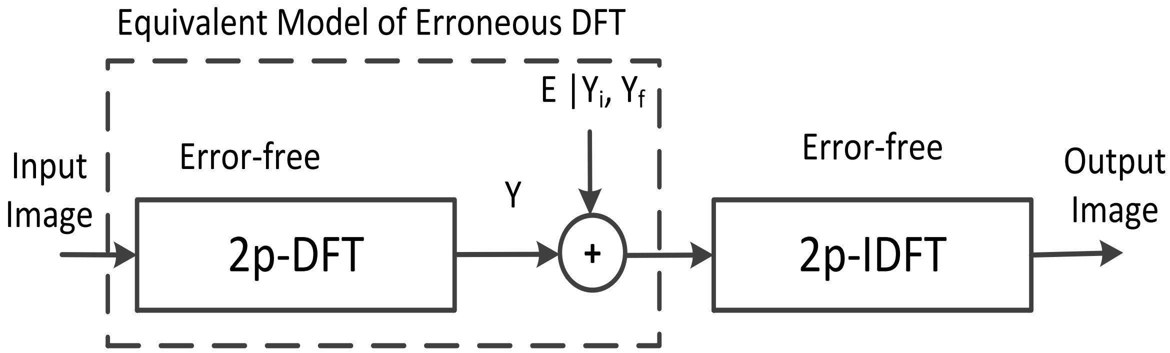


Figure 11. Block diagram of the case study

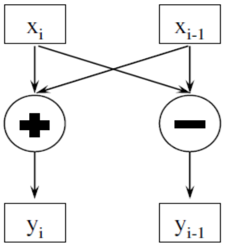
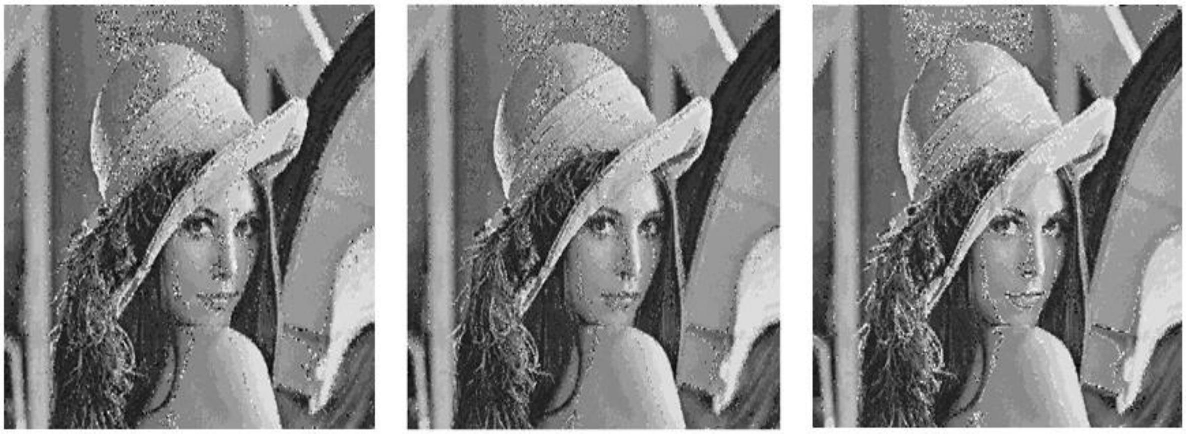


Figure 12. Two point DFT block diagram

To confirm the accuracy of the proposed method, the same input picture was processed through the Monte Carlo method [42], as well as HSPICE simulations, with 32nm technology and



PSNR = 9.94	PSNR = 10.25	PSNR = 10.87
Time = 10 sec	Time = 10 mins	Time = 70 Hrs

Figure 10. Picture quality, PSNR, and timing of HSPICE simulation on right, Monte Carlo in the middle and the proposed method on left.

1V nominal voltage [45]. All tests were performed at a fixed voltage of 70% of the nominal supply voltage (or 0.7V) and a fixed clock period of 100ps. Figure 10 shows the resulting picture quality for the three methods. To mathematically quantify the quality of the pictures, the peak signal to noise ratio (PSNR) of the output pictures was computed and is displayed below each picture. The processing time for each method is also indicated. The proposed algorithm achieves a speed gain in processing time by a factor of 60 and 25,200 as compared to Monte Carlo [42] and SPICE respectively. Alternatively, the proposed model gives a slightly more pessimistic PSNR result, which is appropriate in case the application designer needs to come up with error mitigation algorithms at the application layer (e.g. median filters). In this case, there is a higher likelihood that these mitigation algorithms will improve the image considerably when implemented.

V. Assumptions and Limitations

The model proposed here is calculated based on the assumption that the variation in the threshold voltage of the transistors will result in the gate propagation delay distributions with Gaussian distribution. Even though the concept of the proposed model has the capability to be used for non-Gaussian distributions, the equations presented here are based on the Gaussian distribution assumption. In theory, one can further increase the accuracy of the proposed model by incorporating the non-Gaussian PDFs and propagate the analytical distributions throughout the circuit.

CHAPTER III: IMPLEMENTATION

I. Introduction

Adders have typically been considered good representatives of combinational logic circuits to evaluate DVS effects due to their long carry chains. Many efforts can be found in the literature that characterize the performance of different types of adders under voltage scaling [46]. However, adders are traditionally simulated for a random set of data, and the input dependency of the circuits and timing delays are not considered. Therefore, the characteristics of different applications with the correlation between the input states have not been exploited. In this work, the analytical method proposed is used to demonstrate its application in optimizing the energy error performance of dynamic voltage-scaled adders. In this example, the proposed model is used to switch dynamically between a ripple carry and a carry select adder depending on the performance needed.

II. Ripple Carry Vs Carry Select Adder

In this work, the analytical method proposed is used to exploit the input vector statistics to reconfigure a circuit based on the current operational conditions and optimize the power performance based on the block diagram shown in Figure 13. The implementation is

Error! Reference source not found..

Figure 13. Proposed reconfigurable design methodology.

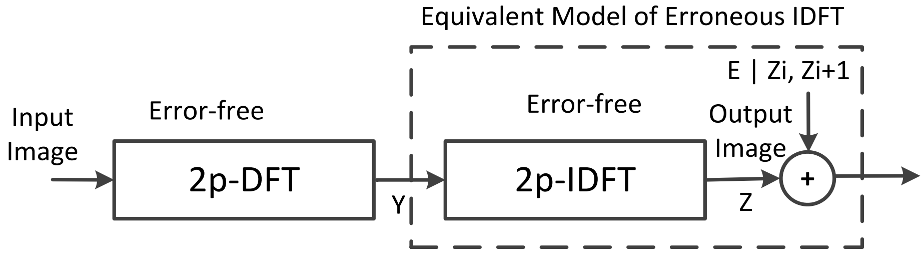


Figure 14. State dependent error model for reconfigurable 2p-DFT in an image compression technique

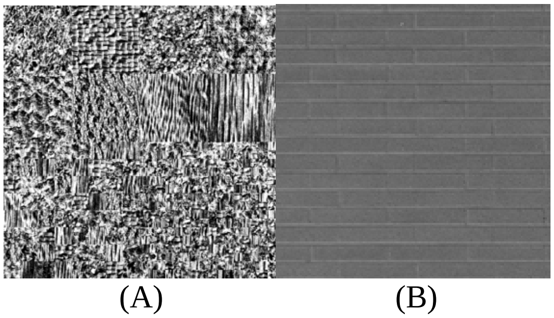


Figure 15. Images with different features

demonstrated on a reconfigurable adder that can switch between a carry-select and a ripple-carry adder within the context of a 2p-IDFT architecture. It is shown that each adder architecture has different tradeoffs that are highly dependent on the input picture statistics.

To better quantify the dependency on the input statistics a simulation system was set up, as described in Figure 14 and simulations were performed using two images with different characteristics. The test images are shown in Figure 15 where the left image (A, random noise) has frequent high variations between pixels while the right image (B, brick-wall) has less frequent changes. Considering image Figure 15 (A) with higher pixel variations, Figure 16 presents a comparison between the performances of the two adders in terms of the picture PSNR versus the amount of voltage scaling. In this figure, the carry select represents a significantly higher PSNR under all levels of voltage scaling.

However, the two curves show a smaller PSNR difference in Figure 17, which represents the same PSNR vs. voltage graph for image in Figure 15 (B) with fewer pixel variations. Comparing the two graphs, the lower frequency of variations in the inputs presented a smaller functional load to the adders. Therefore, the extra hardware in the carry select has not been utilized as efficiently as the hardware permits. This has caused the PSNRs of the image with less frequent variations to have closer values.

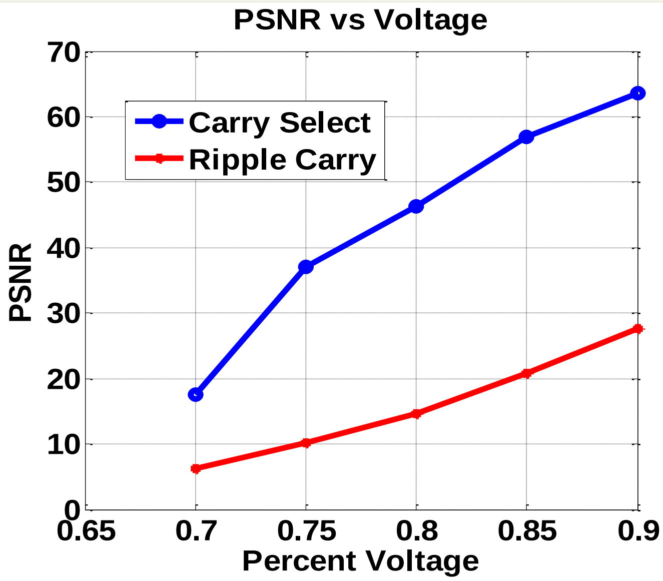


Figure 16. PSNR versus supply voltage for different adders of image at **Figure 15**(A).

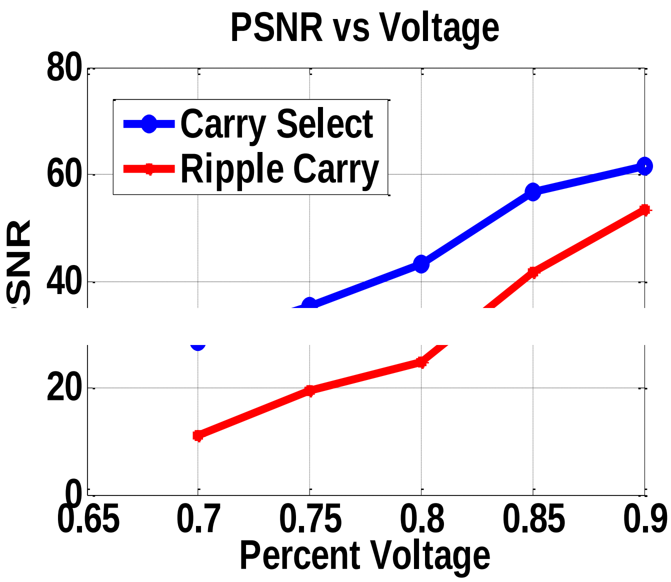
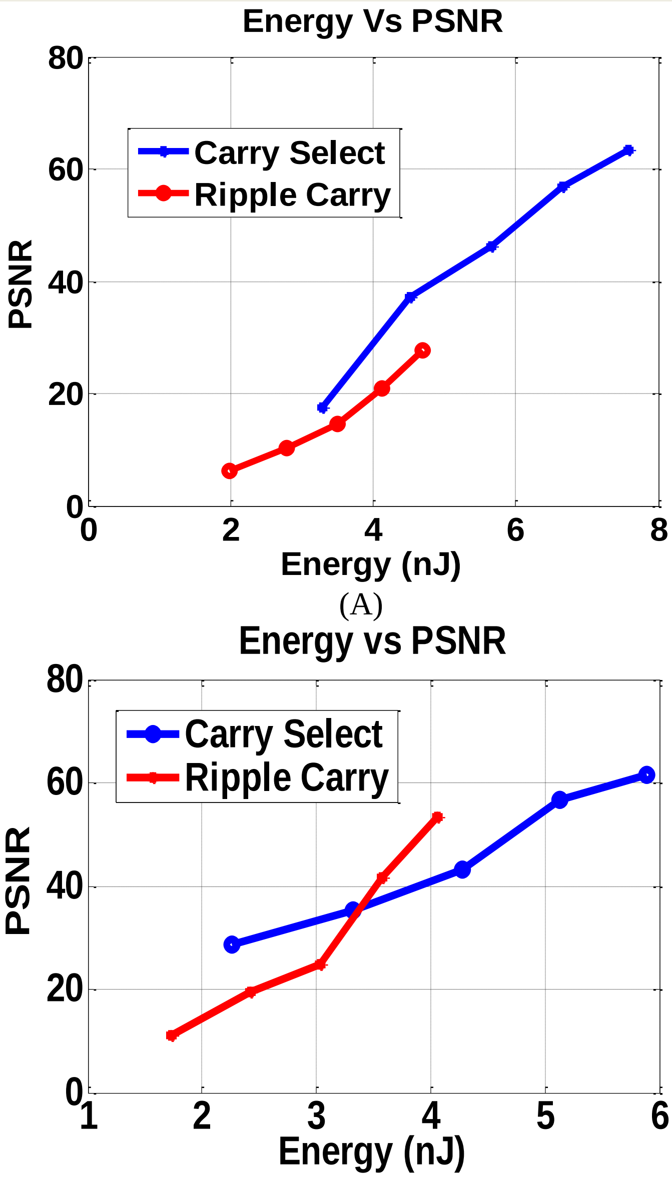


Figure 17. PSNR versus supply voltage for different adders of image at **Figure 15**(B).



(A)

(B)

Figure 18. PSNR versus energy for (A) image A and (B) image B.

Furthermore, Figure 18 shows a comparison between the performances of the two adders versus their energy consumption for the two images shown in Figure 15. For the image depicted in Figure 15 (A) (high frequency of input variations), the carry select performs better for any given PSNR value shown in Figure 18(A). However, when considering the image in Figure 15 (B), depending on the required PSNR, the ripple carry adder presents better PSNR-performance points for higher energy consumption as shown in Figure 18(B). If the input vector is constantly changing

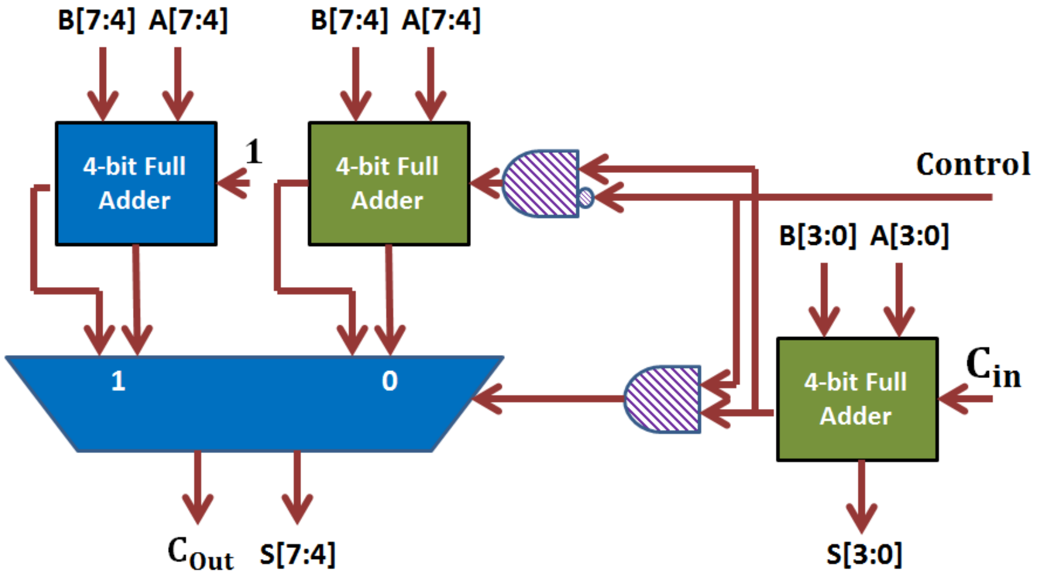


Figure 19. Reconfigurable carry select-ripple carry adder. The green (light) color represents the circuitry for the ripple carry adder. Blue (dark) color represents the extra hardware needed to configure a carry select adder and purple (shaded) color is the controlling gates.

(i.e. Figure 15(A), the carry select will be a better choice since it can calculate the results in a shorter time. Then again, in the case with less frequent changes in the pixel values, the extra hardware of the carry select architecture will not be utilized effectively to justify the extra static energy used. Therefore, the ripple carry will show a better PSNR for the same energy consumption.

III. Simulation Setup

As a case study for the proposed model and to demonstrate its ability to save energy, a reconfigurable 2p-IDFT structure is presented where the adder that implements the 2p-IDFT is depicted in Figure 19. The control signal in the figure will determine whether the adder is configured as a ripple carry (control equal to zero) or a carry select (control equal to one). When used in ripple-carry mode, operand isolation can be applied to the input of the leftmost adder to eliminate dynamic power. The IDFT is used to perform an image decompression using an erroneous IDFT while the compression is performed with an error-free DFT as shown in Figure 14. The erroneous IDFT is modeled as an error-free IDFT and an error injector that introduces errors to the system based on the proposed timing model.

In Figure 20, the input statistics block presents a measure of the rate of change of the input vectors. This measure can be readily generated depending on the target application. For example, one way to generate this is via a simple sign bit rate of change counter. For more advanced standards, such as video compression standards, the information can be inferred from readily available measures such as motion vectors, etc. Based on the frequency of change of the input vectors, a controller could then be used to decide on the best hardware architecture for the

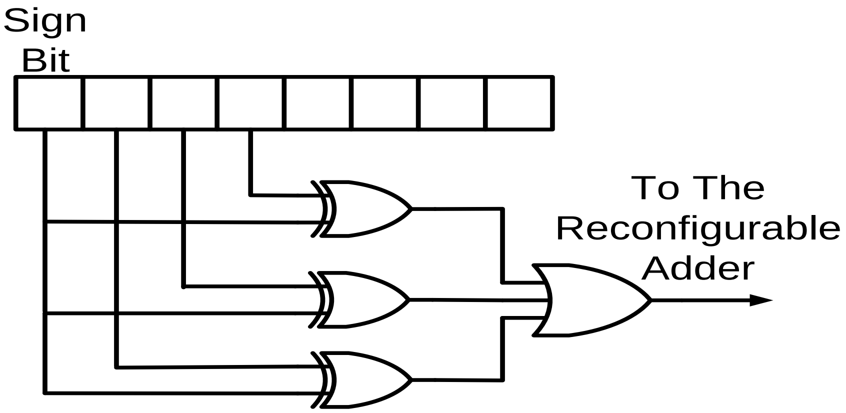


Figure 20. The circuitry for the controller

processing unit. In the case of the proposed reconfigurable hardware, the 2p-DFT is an addition and a subtraction of its two inputs. The subtraction of the image pixels provides a measure of change within the picture. The proposed reconfigurable method uses this measure to decide the appropriate type of adder using the circuit illustrated in Figure 20. This controller circuit uses the result of the DFT subtraction stored in a register as its input. The controller finds the one's complement of the measured value if it is negative with a series of XOR gates with one input connected to the sign bit. Then it checks whether the value is larger than half the register size with an OR gate as shown in Fig 10. This is because the carry select has two segments and the first part of the adders share the same hardware. The output of the OR gate is directly connected to the control signal of the reconfigurable adder circuit.

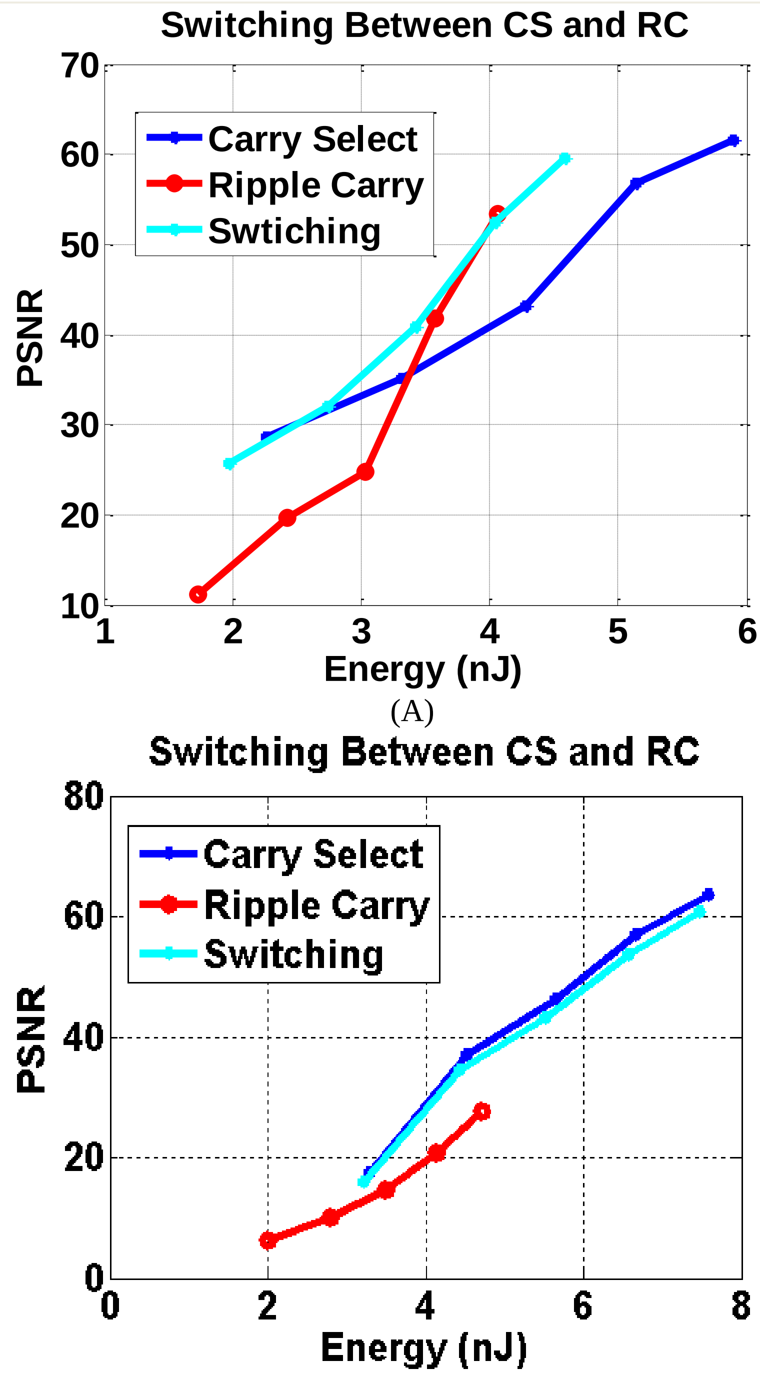
CHAPTER IV: RESULTS

I. Introduction

To demonstrate an application of the proposed timing model in optimizing the power vs. performance tradeoff, a reconfigurable carry-select/ripple carry adder has been proposed. The proposed timing model is used to characterize the PSNR vs. energy performance of the two adders, and it was established that the optimum performance of the adders would be a function of their input statistics, meaning that a certain adder might be the optimum choice under certain input statistics while the other adder would excel under different input statistics. In this section, the suggested circuitry for switching between the adders is used to switch dynamically between the two adders and achieve the power optimized performance.

II. Performance Comparison

The Pareto-optimal choice of an adder based on the input picture statistics can be made by switching between the two adders as the characteristics of the image change. This will stop the carry select from draining too much energy when there is no need for fast computations. Then again, when there is a higher variation in the image the extra capability of the carry select will help to increase the PSNR of the resulting image. The resulting PSNR vs. energy figures for images in Figure 15 (A) and Figure 15 (B) are shown in Figure 21 (A) and Figure 21 (B) respectively. The figures indicate that the image with less frequent changes switching between the two adders will significantly improve the energy performance of the adders. However, in the case of more random inputs, the carry select will remain the adder with the best energy PSNR performance and the proposed switching adder will perform very closely.



(A)

(B)

Figure 21. PSNR versus energy for the DFT with reconfigurable adder for image with higher frequency of changes

CHAPTER V: Conclusion

A fast and accurate analytical, statistical state-dependent model is presented to calculate the timing distributions of the output bits of voltage scaled logic circuits. The mathematical foundation for the model is explained, and its accuracy has been verified. Additionally, an additive statistical error model based on the timing violations has been obtained. Furthermore, to determine the accuracy of the model a case study of a two-point DFT employing the model was conducted, and the output picture quality in terms of PSNR has been compared to HSPICE circuit simulation.

To further demonstrate the power of the model in saving energy and the importance of the input statistics, a reconfigurable design methodology employing the statistics of the circuit input vectors is presented. Based on the proposed timing model, an input-dependent reconfigurable carry-select/ripple-carry adder is designed and demonstrated in a reconfigurable 2p- IDFT to trade-off image quality and energy consumption. The proposed architecture achieves a significantly better performance for the lower variation images and matches the performance of the optimal adder for the images with higher variations.

I. Summary

The challenge of designing systems with optimized energy consumption in the presence of growing variation has been identified. The current practice of overdesigning systems and disregarding variation effects on the system performance or major approximations on the currently available modeling techniques has resulted in inefficient systems and overusing resources. To address this problem, the effects of variations in propagation delays of circuits have been analyzed, and a fast analytical model for the propagation delays have been proposed. The model can provide timing data for a combinational circuit based on the operating supply voltage. The model considers the input state of the circuits as an additional measure to ensure exact prediction of the propagation delays. The proposed model is verified by comparison to Monte Carlo simulation on HSPICE platform. The analytical nature of the model also assures one or two orders of magnitudes faster speed compared to similar modeling efforts. The proposed model can be used to reduce system overhead by releasing the bond by full correctness in error tolerant parts of systems and optimizing the energy vs. quality trade-off. The model is further demonstrated to accurately predict the timing violation errors of a case study voltage over scaled ripple carry adder utilized in the image compression circuitry.

Furthermore, to illustrate the energy saving possibilities with the proposed state dependent model, the model is used in a reconfigurable carry select/ripple carry adder configuration. It was shown in the implementation that such a model is capable of saving energy when it is used to switch between the adders depending on the changes in the input statistics.

The proposed state-dependent analytical model empowers circuit designers to explore different circuits and generate fast estimations for signal delays. The model can also be used to understand the propagation delay failures within the circuit and to improve the uniform utilization of the circuit and available timing. The designer can further compare different design architectures to find the least sensitive to the variation or consider its effects even under extreme voltage or frequency-scaled conditions.

II. Future Work

Future work on the presented research can perform the following:

- incorporate the fan out of the gates in the look-up tables used,
- consider the location of the gate within the die and change the standard deviation of the distribution used according to the relative location,
- development of other error tolerant mechanisms such as communication systems and studying the effects on system-level performance, and
- use of non-Gaussian distribution to further improve the accuracy of larger systems.

PART II: Low Overhead Delay-Based Physically Unclonable Function

CHAPTER VI: Overview

I. Introduction

Rising incidents of counterfeiting electronic chips have posed a threat to the semiconductor industry. Counterfeit ICs typically lack the performance and material quality but are sold as genuine and could result in serious reliability issues as well as reputation or revenue loss. Additionally, security has emerged as a significant parameter in IC design and designers are required to consider the possibility of reverse engineering, counterfeiting, and tampering attacks. The current best countermeasure when verification is possible is to verify the authenticity by producing a number of secret keys only known to the verifier and the genuine party. The key is then encrypted and placed into nonvolatile EEPROM or SRAM with a supplying battery. These secret keys securely authenticate the parties communicating. However, these approaches have proved to be costly due to the design area used. Also, manufacturing a tamper evident memory to preserve such a key is a challenge and will have a high power consumption. This power is drained to keep the data and to provide mechanisms to detect and prevent attacks. Moreover, such systems are vulnerable to a potential communication intercepting attack. Physical unclonable functions (PUFs) have recently emerged as an innovative measure to address this issue by providing a vector space of the keys and mapped responses through the use of an unclonable chip. PUFs map the random variations in silicon manufacturing technology from device parameters such as channel length and threshold voltage to circuit level randomness. In these PUFs, a repeatable key can be generated from the random digital response of the PUF circuit to a challenge bit stream without it ever being stored into any memory. In this work a simple, efficient and low overhead delay-based

PUF is presented. This PUF is based on the random characteristics of the setup and hold time violation of FFs, which fluctuate with variations in the fabrication process. Use of this randomness in multiple stages will ensure a key response vector space that is diverse enough to be unpredictable and it is unique to the specific chip used.

II. Physically Unclonable Function (PUF)

Physically unclonable functions (PUFs) are easy to fabricate die-specific random functions that are widely investigated as a solution for many security concerns. A PUF can be modeled as a function with unknown internal parameters returning a response for challenges arriving as an input to this function. Each PUF is virtually unclonable even with known design layouts and development procedures. This unclonability originates from the randomness and unpredictability of device characteristics during manufacturing. PUF reliability is due to the difficulty of measurement on internal signals, an estimation of the performance of manufacturing two exactly identical chips. Alternatively, a physical attack to PUFs has the possibility of changing the sensitive internal characteristics of the chip and, therefore, changing the challenge-response vector space produced by the specific chip [47]. Likewise, duplicating PUFs would need to reproduce the hardware with the exact same behaviors, which are generated by the specific fluctuations happened during the manufacturing process. Additionally, because the information is stored in the physical parameters of the device, there is no need for the PUF to stay on continuously. This will reduce the probability of a successful attack as well as a reduction in the power consumption. Furthermore, a PUF can generate a very large number of random and difficult to predict challenge response pairs [48][48]. Having a vast number of random key-response pairs for a low cost makes it practically infeasible for the attacker to predict the next key that would be used in the next authentication.

One of the main measures of the performance of the PUF is the inter-distance and intra-distance [50]. The inter-distance is the difference between the responses from two different PUFs to a single challenge. The intra-distance is the difference between responses of a single PUF to the same challenge in multiple instances under different environmental conditions. The challenge used for both metrics should be the same in these calculations. The measurement of the distance will be dependent on the type of challenge response used but in the case of bit streams, hamming distance is typically used. An ideal PUF would return two non-related responses (50% hamming distance) for the application of the same challenge to two copies of the PUF and return the same response as its intra-distance [50]. However, practical PUFs normally return an array of close to 50% hamming distance responses making a histogram of correlations. One of the limitations of the PUF is also that a practical PUF will typically deviate slightly in its intra-distance with the changes in the environment, which is a factor in some applications of PUFs.

III. PUF Applications

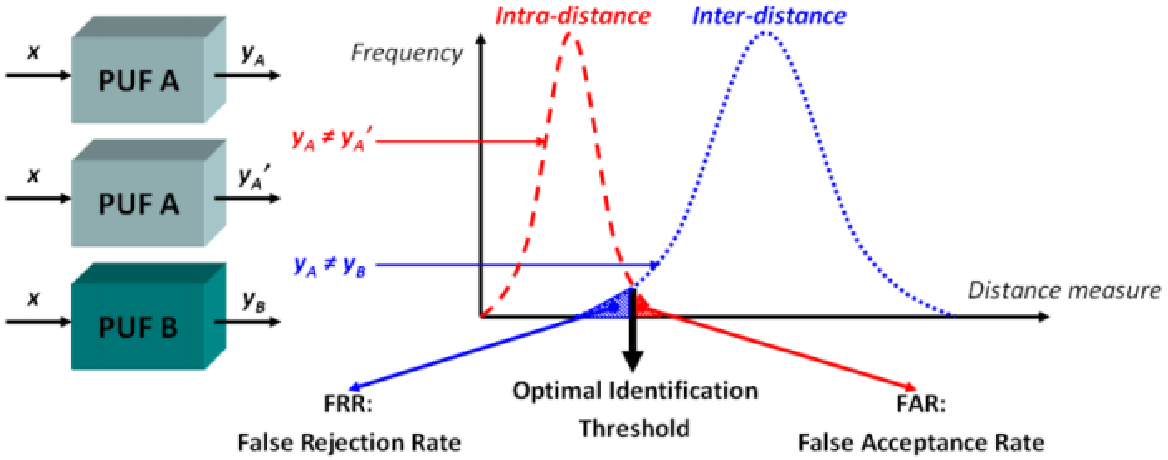


Figure 22. Inter and intra distributions and determination of the detection threshold [50]

PUFs are attractive in many applications mostly with a focus on security or authenticity. Their applications are also expanding as more implementation methods are used. In addition, the suitability of a PUF for an application is mainly dependent on the requirements of the application. As an example, PUFs used for a secure communication must have a lower error rate while PUFs used for random number generation can be relaxed on the error rate and use nosier responses [47]. PUFs have specific applications in FPGAs intellectual property (IP) protection, software licensing, smart credit cards, securely running a code with authenticity requirements, hardware-specific software licensing, generating random numbers for other cryptography applications, and protecting the authenticity of devices. Some of the main applications of PUFs are discussed here.

A. Signature Generation

Due to the unclonability property of PUFs, they have been used in anti-counterfeiting technologies to generate an exclusive signature for authenticating devices. The response of a PUF can be used to identify a device similar to the way biometrical information is used to identify people. Since the signature generated by PUF cannot be duplicated and is also tamper resistant, PUFs have found applications in authenticating and intellectual property protections. The basic idea to use a PUF to generate a signature is the following. The responses of the PUF to many challenges as well as the identity of the device it would be used in are stored in a database during the characterization phase. To identify a device during the identification phase, the PUF is tested with one of the challenges. If the returned response is close enough to the expected response stored in the database, the identification is successful. Given that PUFs typically have a vast number of challenge-response pairs, to ensure the communication cannot be overheard by an intruder and repeated, the challenge response pair that is used will be permanently deleted from the database. The decision for the closeness of the response of a PUF to the database stored response will depend on the performance of the specific PUF. If the histograms of inter-distance and intra-distance of the PUF do not overlap (meaning that a correct identification is always possible), a threshold limit would be decided that is between the two histograms. However, if it does overlap, then there would be a tradeoff between a false acceptance and false rejection depending on where the limit would be placed. In this case, the threshold is typically placed at the intersection of the two histograms [50].

B. Cryptography

Many encryption algorithms use a secret key to encrypt the data that needs to be communicated. This key will need to be stored in a memory on both the encrypting and decrypting sides. However, storing this key in a non-volatile memory and ensuring its security is a challenge for these applications. PUFs can be used to generate the key as a response to a fixed challenge with some modifications to ensure the key is uniformly random and perfectly reliable [51]. These modifications are done in a two-phase algorithm. During the initialization phase, the response of the PUF is an algorithm to produce an output called helper data. The response together with the helper data are stored in the verifier database but not in the device that needs to be verified. In the verification phase, the PUF that is placed in the device will be presented with the challenge and the helper data. If the device contains the right PUF, the algorithm would be able to use this information to obtain the key used in the communication. It has been shown in [47] that even if the helper data would be revealed, the information will stay secret. The advantage of PUF-based cryptography is that it does not need to store this key anywhere in the device to be verified either in volatile or non-volatile memory, which offers a full security against attackers trying to copy the content of the memory, which is especially interesting for the embedded devices. Furthermore, physical attacks to the PUFs are typically not practical since any physical change in the PUF hardware can trigger changes in its sensitive circuitry and change its response.

C. Random Number Generation

Since the PUFs generate their output based on being non-correlated, the response of a PUF with some modifications can be used as a source of randomness. Additionally, because this randomness is inherent in their manufacturing variability, no non-volatile memory is required to store their output, which offers an additional security in cryptography applications. Truly random numbers have been generated in many implementations of PUFs including SRAM PUFs, ring oscillator PUFs, and D-flip-flops [53] [54]. Different implementations of PUFs will be discussed in the next section.

IV. PUF Implementation

There are a variety of circuits, materials, and methods to construct PUFs. They can be categorized based on their construction and operational basics such as non-electronic or electronic PUFs. The non-electronic PUFs are those that extract their randomness from a non-electronic nature even though they might use electronic equipment in other parts of their design. Examples of these PUFs are optical PUFs and paper PUFs. Optical PUF [55] is one of the earliest versions of unclonable functions constructed based on the random reflection of the light from glass spheres in a transparent material. The reflection is captured by a camera and digitally processed to authenticate the device [56] [57]. Paper PUFs were first introduced as an anti-counterfeiting method before the introduction of the PUF concept. It is created based on the random structure of papers [58]. Among the electronic PUFs, using a randomness inherent in the silicon manufacturing has proved to be an attractive method to generate a secret. A silicon PUF, as these PUFs are called, is based on the idea that even though the layer masks and manufacturing process of all the samples of a chip are identical, there is a slight variation between different samples of them. Silicon PUFs transfer this variation to generate random bits and store these bits as long as the chip lasts. Additionally, because of the uniqueness of the manufacturing process, it is impossible to generate a second copy of the same exact chip even using identical masks and procedures.

Several implementations of these PUFs have been proposed over the years, which can be categorized into weak or strong in terms of the number of challenge response pairs. Electronic PUFs also mostly fit into two categories in terms of their implementation method, state-based or delay-based. Most electronic PUF implementations magnify the alterations in variations of characteristics such as threshold voltage, path delay, conductivity and frequency of oscillation. The next section will briefly review the major PUF implementation methods and their advantages and disadvantages.

A. State-Based PUF

A digital memory cell can settle to two stable states, i.e. a zero and a one. Each cell can be forced to choose one of these two states to store information. However, if the cell is brought to a transitional state, it can settle into either one of the stable states. Furthermore, it has been shown that the state that a digital cell chooses is not a function of the circuit implementing it but rather it is a function of the internal physical mismatch of it, which itself is caused by manufacturing variability of the transistors. For this reason, a digital memory cell is a good candidate to create a state- or memory-based PUF. SRAM, butterfly, latch and flip-flop PUFs are examples of the state-based PUFs that will be discussed shortly here.

B. SRAM PUF

One of the simpler implementations of PUF is through the random power up state of a static random access memory (SRAM) circuit before the execution of any write instructions [48] [59] [59]. The SRAM circuit structure consists of two symmetrical inverter halves. However, the circuit does not determine the startup value of the stored bit. This state-based implementation is based on the idea that the bits of a SRAM will choose random values at power up due to their manufacturing variability and their initial states, meaning that some cells of the SRAM would prefer zero and others one for their initial state. This phenomenon can be used to generate the fingerprint of these PUFs since it is experimentally proved that most SRAM cells have an explicit preference of choosing one state over the other. Moreover, Mukhopadhyay et al. [60] have shown that there is enough variation in the power up state of the SRAM cells to be used as a source of randomness. Additionally, the distribution of cells with zero and one preference is random. However, because of the limited size of a SRAM, the challenge-response space of SRAM-based PUFs is limited by the die area. Known as weak PUFs, these PUFs are typically used as a seed for another cryptographic algorithm to generate a larger key-response space.

C. Butterfly PUFs

Implementing SRAM PUFs in FPGAs is generally not possible. This is because on most FPGAs the SRAM cells are difficult reset to zero right after power-up. Therefore, the randomness in the SRAM bits is all erased. Another issue with implementing SRAM PUFs on FPGAs is that the power up process is needed to get the random bits, which is not always possible. To address these issues, butterfly PUFs have been proposed by [62], which is based on mimicking an unstable condition of a SRAM bit with two back to back connected flip-flops. In addition, the preset and

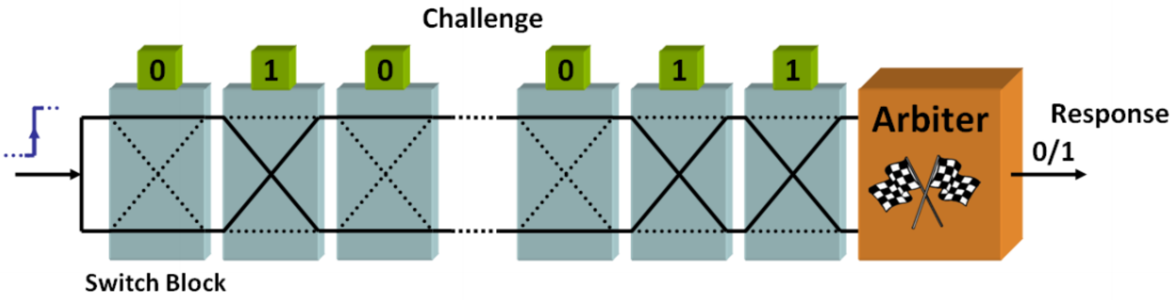


Figure 23. The concept of the Arbiter PUF [50]

clear signals of the flip-flops are used instead of a hard-reset or power-up of the cell to introduce an unstable state.

D. Sense Amplifier PUF

Another state-based implementation of PUF is through the use of sense amplifiers (SA). SAs are clocked circuits that sense the small differential voltages and amplify them to a zero or one level. The SA is typically used in the read path of a memory cell to sense and refresh one bit. Each SA cell has a unique internal offset due to manufacturing variability. This specific offset is used to generate one bit of response for the PUF. However, a practical sense amplifier might have a tendency to have a bias towards generating one of the logic levels with a higher probability. This issue can be improved by providing the SA inputs with voltage differences higher than the offset voltage. However, some of the bias voltages are extremely small, which would cause the PUF to have unreliable response bits, which is a major issue with the use of these PUF sets.

E. Delay-Based PUF

The majority of PUF implementations are delay-based PUFs. This type of PUF is named so because they are based on measuring random variations on the delay of logic or interconnects. The main concept used in these PUFs is to compare the delays of identically or symmetrically designed structures and exploit the propagation delay differences introduced due to the manufacturing variation. Multiplexers and edge-triggered flip-flops are very commonly used in different configurations to detect the first signal that arrives at a checkpoint. Arbiter PUFs and ring oscillator PUFs can be named as some of the main implementations of delay-based PUFs, which are reviewed in the next section.

F. Arbiter PUF

An example of a delay-based PUF is an arbiter PUF, one of the first published PUF papers introduced by Gassend et al. [63]. In this implementation using two paths of interconnected multiplexers, a race condition has been created with a so-called arbiter circuit deciding the winner. Since the two paths are designed identically, the winner is not known by design [48]. During manufacturing, the variation will determine which path will have the least propagation delay, which generates the randomness. However, if the propagation delay difference would be less than the setup and hold time required to generate a definitive result, the variation in the arbiter circuit or noise can influence the results, which can force the arbiter circuit into metastability and introduce noise in the response of the PUF. Figure 23 illustrates the concept of the arbiter PUF. The first implementation of these PUFs uses a two-input two-output circuit named a switch block, which can pass the inputs directly to the output or reverse them and pass them depending on the challenge, which can control its mode. The input of the interconnected arbiters is a positive signal edge, which propagates through all switching blocks in two parallel paths and generates a unique bit at the output of the arbiter. In the case of the arbiter being a D-flip-flop and switch blocks being multiplexers, one of these paths ends at the D input of a DFF and the other at the clock with the challenge connected to the select of the multiplexers. To generate a larger response, multiple identical structures would be used to generate as many response bits as needed. It was proved later that the propagation delay is additive by nature, so the final propagation delay is the sum of all path delays. Therefore, so-called model building attacks [64] showed that with having a limited number of challenge-response pairs the delay of the gates could be calculated and the rest of the vector space can be revealed to the attacker. The subsequent efforts of addressing this issue such as feed-forward arbiter tried to use some of the response bits for the switch blocks. However, more advanced modeling attacks were still successful in generating the delay models of the paths and uncovering the challenge response pairs. Later improvement methods added difficult to invert input and output circuits to control the PUF, but these have also been proved breakable and have also increased the area overhead and the delay of this type of PUF.

G. Ring Oscillator PUF

Another example of a delay-based implementation of a PUF uses a different method to exploit the variation in the gate delays [65] [66] [67]. A ring oscillator structure is created by feeding back the output of an odd number of inverters to the first input resulting in an oscillation. The frequency of this oscillation will directly depend on the delay of the inverters used. Therefore, one could measure the frequency of the oscillation to know the exact addition of the delays in the path, which has a slight variation due to the effects of the manufacturing process. To measure the frequency, a counter is set to count the number of signal edges within a given time period. However, unlike the arbiter PUF that uses two parallel paths and offers some resistance to noise the ring oscillator is more susceptible to frequency change due to noise. Therefore, multiple subsequent measurements are made to average out the readings and ensure more accurate reading [68]. To create an array of response bits, N identical ring oscillators are used to generate N slightly different frequencies. The challenge will be used to select two of these frequencies and compare them. In comparing each two pairs of frequencies, one output bit of the PUF is generated. If the frequency of the first oscillator is higher, the response would be a zero. Otherwise, it would be a one. With using N different frequencies, there are N (N-1)/2 possible pairs and log (N!) bits available. For every sample of this PUF, the frequency of the ring oscillators is fixed after manufacturing and, therefore, the response bits will be fixed. In practice, the ring oscillator frequencies vary with the environment noise sources. Another measure to reduce the noise in this scheme, which is suggested by [69] is to find the most stable bits of 8 frequency comparisons to represent the response bit. This ensures the effect of noise on the close frequencies will be minimized.

H. Scan PUF

Among the delay-based PUFs with a vast challenge-response space and minimum area overhead the scan PUFs can be named [70] [71] [72] [12]. These PUFs use the scan chains built into the FPGAs to implement a PUF using the variation in the delays of these chains. A modern FPGA typically holds anywhere between hundreds to millions of these chains. Only a small portion of these chains would be required to implement these PUFs with high randomness. In addition, the area overhead of these PUFs is minimal due to the use of the existing hardware and the only extra area usage in these PUF are controlling circuitry. Because of the same basic principle of using the existing hardware, they have some inherent insecurity. Specifically, the tester must be able to access these chains, and if the access method used by the tester is not limited, some security concern can be raised. Furthermore, these PUF are not intrinsic to other chips, as they need FPGA scan chains.

CHAPTER VII: Implementation

I. Introduction

The main ingredient for generating a PUF is identifying a source of a repeatable randomness. The silicon manufacturing process has proved to be an easy to access source with its natural variations. Many methods have also been proposed to extract this randomness from the device level to the circuit level. One of the simplest forms that this variation is observed is through the normally distributed propagation delay of logic gates. This analog delay time needs to be converted into a digital alteration to have a PUF with digital challenge and response. One of the simplest circuits that can perform this conversion is a D-flip-flop. In this work, the combination of multiple series states of FFs and XORs are used to take advantage of the random variation inherent in silicon manufacturing and implement a delay-based PUF. The next section will further explain the details of this implementation.

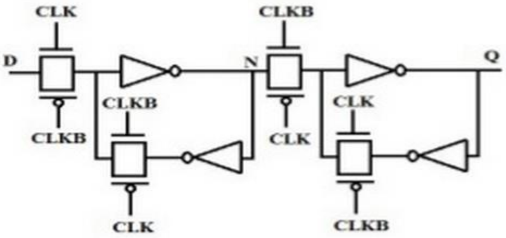


Figure 24. Schematic of a DFF

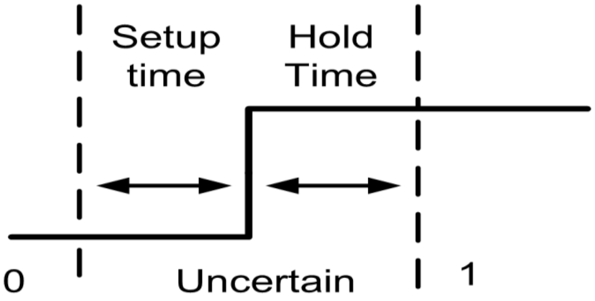


Figure 25. Uncertain region with respect to the clock

II. Process Variation & D-Flip-Flop

Transistors manufactured in a die are never perfectly identical. In fact, with further reduction of the gate lengths, the variation on the generated transistors further increases. This variation will result in current conductivities that are not uniform. Alternatively, D-flip-flops, as shown in Figure 24, consist of two latch or switch and hold sections. Variations of the current that the switches can pass through as well as the impedance seen by these switches directly affects the speed they can charge or discharge their input and output capacitors. This will result in variations in the FF's output if there is a transition within the sensitive time frame that the signal is being sampled. This sensitive time, which extends from before the introduction of the clock and continues until after its arrival, is shown in Figure 25. The sensitive period of time before the clock is named the setup time and the time frame after the clock arrival is named hold time. If the input of the FF has a transition within this period, the output of the FF will depend on the specific characteristics of the transistors of the FF used. In a normal use of FFs, the system will be designed to ensure these uncertain regions are avoided. This allows an error-free system to perform exactly as it is expected. In this work, however, the output randomness generated with this method is used as a source to generate a secret specific to the hardware being used. Since reducing the operating voltage increases the charge or unchanging times, dropping the supply voltage in a FF will cause the expansion of the setup and hold times. To further encourage the uncertainty for the PUF application of the FF, the supply voltage is also reduced, which will be explained in greater detail in the next section.

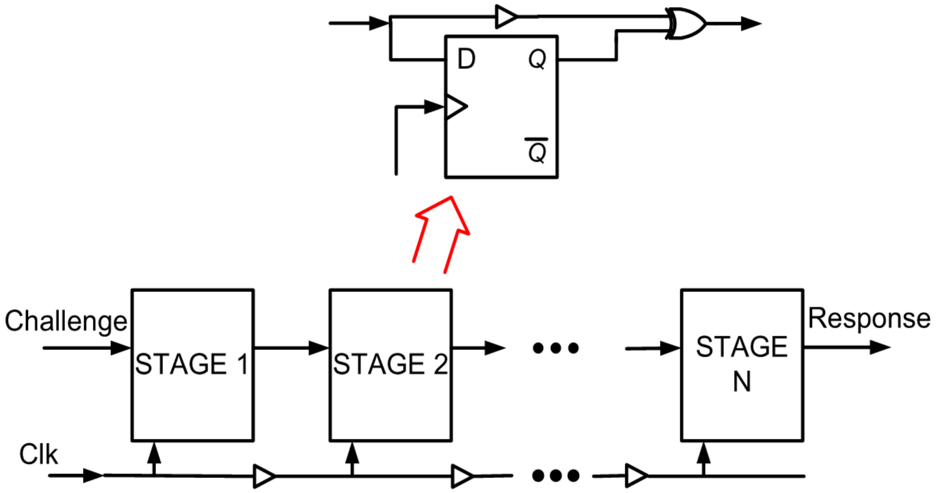


Figure 26. Schematic of the proposed PUF

III. Principle of Operation

In this design, we frequently violate the setup and hold times of FFs by using two distinct non-multiplicand frequencies for the data and the clock of the FFs. When the data frequency is much higher than that of the clock, the transition edges of the two signals will have close to random distributions with respect to each other. In effect, this will test the FFs uncertain values for randomly selected time intervals between the clock and data edges and produce a unique to the hardware bit.

Figure 26 shows the schematic of the proposed PUF. In this arrangement, the challenge enters the first FF, and the response is produced at the output of the last FF. depending on the ratio of the total uncertain time of the FFs to the period of the challenge, a portion of the challenge bits would be replaced with new uncertain bits at each stage. A clock frequency of 100MHz and the challenge frequency of around 2.28 GHz have been used for this design. It was practically tested that fifteen stages of the identical circuitry will ensure that there would remain no portion of the time where the input challenge would pass through without enough shuffling. Furthermore, adding a XOR gate at the output of each stage will serve two vital purposes. First, it will guarantee that the frequency of the data input of FFs will remain high and will not sync to the clock frequency, which is important to keeping the ratio of uncertain regions to the data period and producing uncertain bits at later stages. Second, it will block the characterization attacks attempting to identify the behavior of each FF by further mixing the output of each stage with the previous outputs. Characterization attacks listen to a small sample of the challenge response pairs and try to characterize the performance of each part of PUF. This has been proved viable in the initial structures of arbiter PUFs, and they have been successfully characterized. Therefore, it is vital to block any attempt to explore the internal behavior of the PUF and only present the final response to the user. Insertion of buffers in the data and clock paths also assists in matching the delays of the FFs and XOR gates. This will make the system more robust and improves the sustainability of the design with the introduction of the environmental noise and temperature variation.

IV. Challenge-Response Space

PUFs can be divided into two categories of weak or strong PUFs based on the size of their challenge response pairs [48]. Having a larger space is an advantage in PUFs since it will ensure the uniqueness of the challenge that the verifier provides and no need for repeating a challenge. A strong PUF also offers better security since the challenges cannot be tested beforehand and used for an attack. Moreover, in the proposed PUF, the basic principle of the implementation is that each FF can be tested for any time difference between its clock and input signal within its sensitive region. Since the time difference is an analog measure, there is a virtually unlimited number of tests that can be performed. This results in one of the key advantages of the proposed PUF, which is a vast challenge response pair and characterizes it as a strong PUF. In fact, any sequence of zeroes and ones can enter the proposed PUF and would be a viable challenge. Furthermore, except the increase in the intra-die distance in the responses with very small sizes, the length of the selected stream can be adjusted based on the security need. This limitation is because of the percentage vise importance of one-bit flip due to environmental effects to the total length of the response. In other words, if the selected response is too small, if a single bit flips due to the environmental or noise effects it would be a bigger percent of the response and can make the response unverifiable.

CHAPTER VIII: Evaluation

I. Simulation Setup

The proposed PUF has been simulated using the HSPICE simulator with a 45nm technology node. Each transistor’s gate length, width, and threshold voltage have been varied to simulate the effects of the process variation on transistors. The simulation has been performed with 15% global and local variations in the width and length of the transistors using a uniform random number generator. The global deviation from the average reported in the characteristics of the technology files has been selected once per every new chip and the local variation has been added or subtracted on each transistor. The threshold voltage has also been varied using a uniform distribution of the global variation with 10% deviation per chip. The local variations have been added using a Gaussian distribution determining the deviation from the average globally varied threshold voltage. Furthermore, these variations have been performed separately for the N-type and P-type transistors with the assumption that there is little effect from the global variation of each type to the other. In addition, a clock frequency of 100 MHz and an initial input or challenge are modeled with voltage sources. The clock and its inverse have used a pulse voltage source with 2-picosecond transition times and 50% duty cycles. Alternatively, the challenge has used a pseudo-random bit generator voltage source. This voltage source in HSPICE uses a linear feedback shift register (LFSR) with the taps determining the specific challenge being used. The frequency of this source is also 2.2815569649GHz or approximately 22.8 times the clock frequency. This is to ensure that there would be many chances for a signal edge to fall within the setup and hold times of the FFs. Also, it is for avoiding the two frequencies being a multiplication of each other for enough number of cycles that the challenge is generated. Additionally, inter-die hamming distances have been measured at 25-degrees centigrade while the supply voltage has been dropped

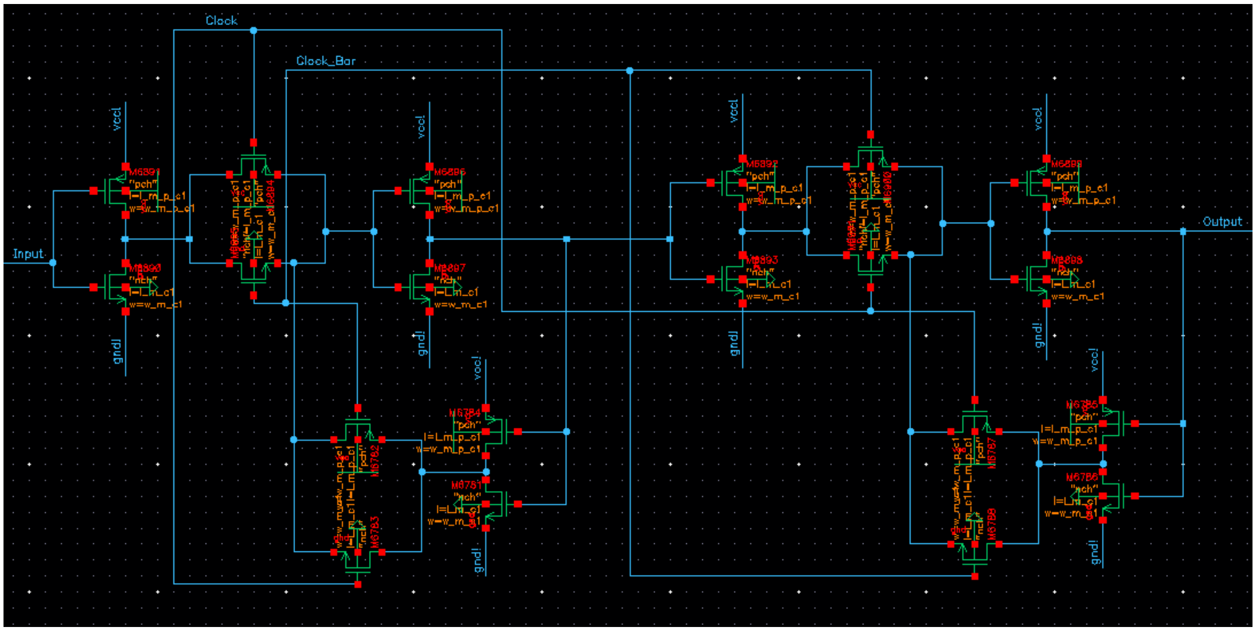


Figure 27. D-Flip-Flop implementation used

to 80% of the nominal voltage to further extend the propagation delays and the setup and hold times.

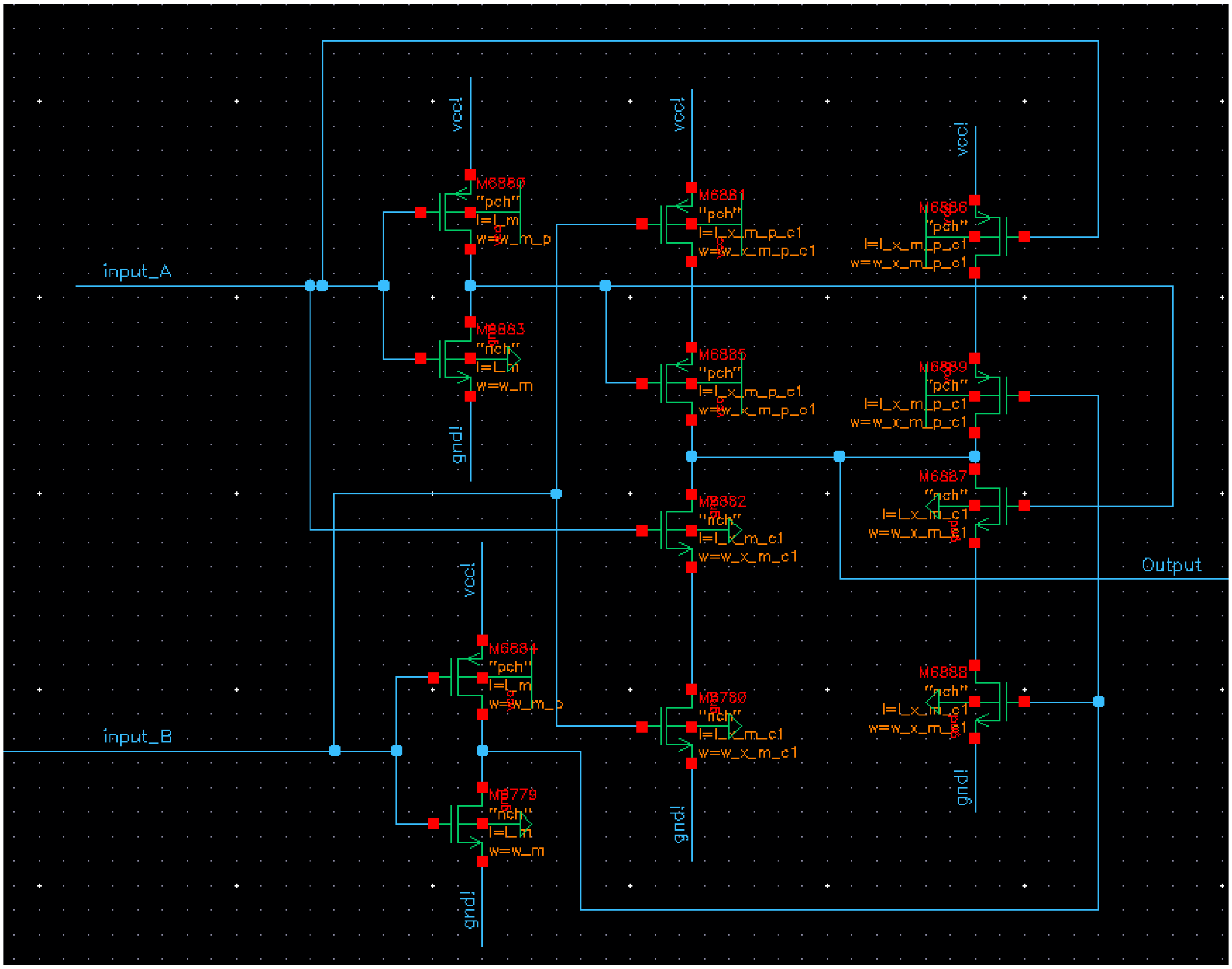


Figure 28. Schematic of the XOR used

II. Performance

The performance of the proposed PUF has been characterized in three categories of uniqueness, uniformity, and reliability. The hamming distance, which is the percent of bits differing between two streams of bits, is used as the measure of the difference between responses.

A. Uniqueness

An ideal PUF is one that generates an average of 50% hamming distance between the responses of chips meaning that half of the bits of the responses would be different. This is because a bias in the responses can be exploited by attackers and used to predict the future responses. If a

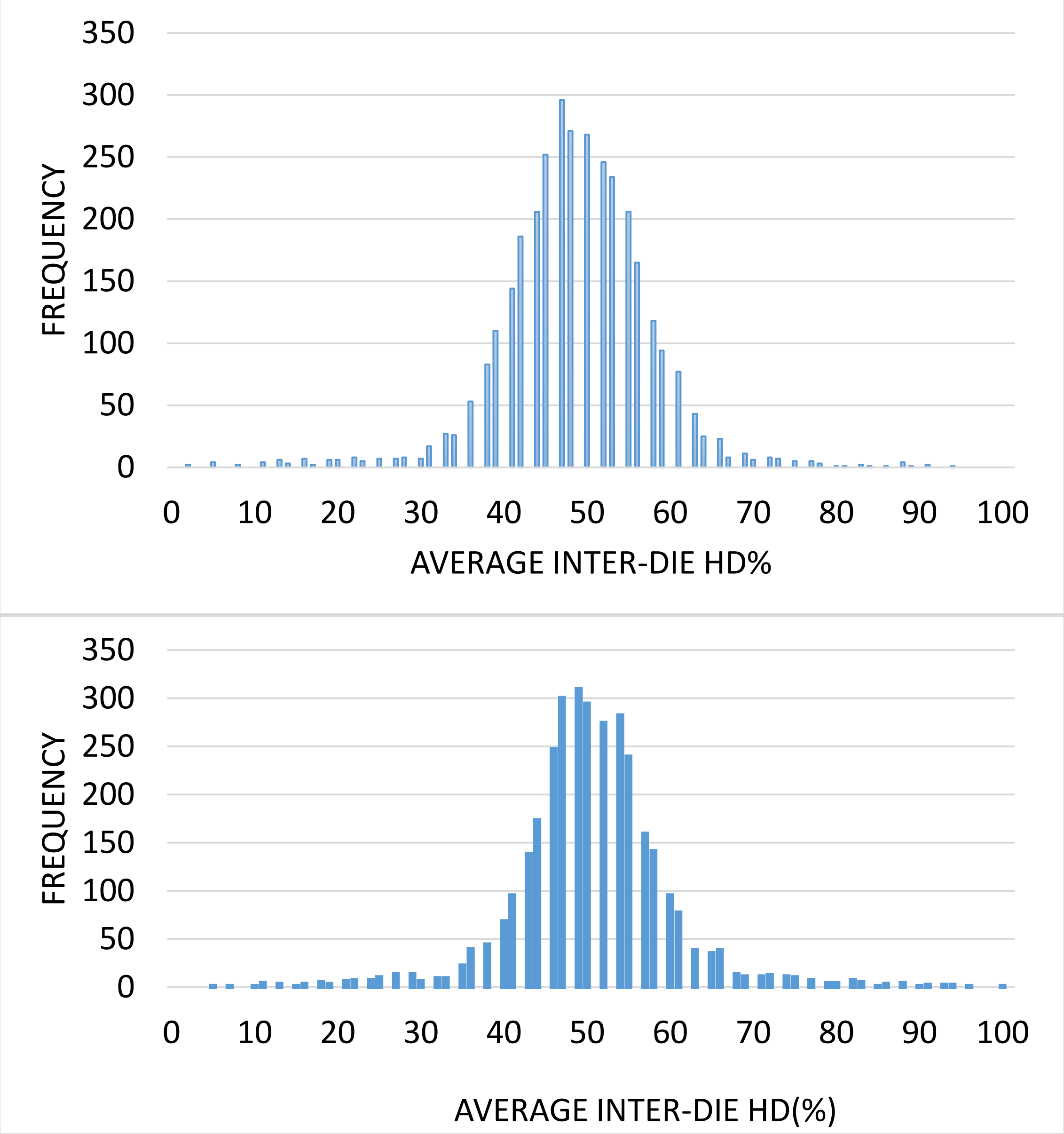


Figure 29. Average Inter-Die Hamming Distance for different challenges

PUF’s uniqueness does not reach close to 50%, then some post-implementation algorithms have been proposed to improve this performance. However, this extra step will add the possibility for

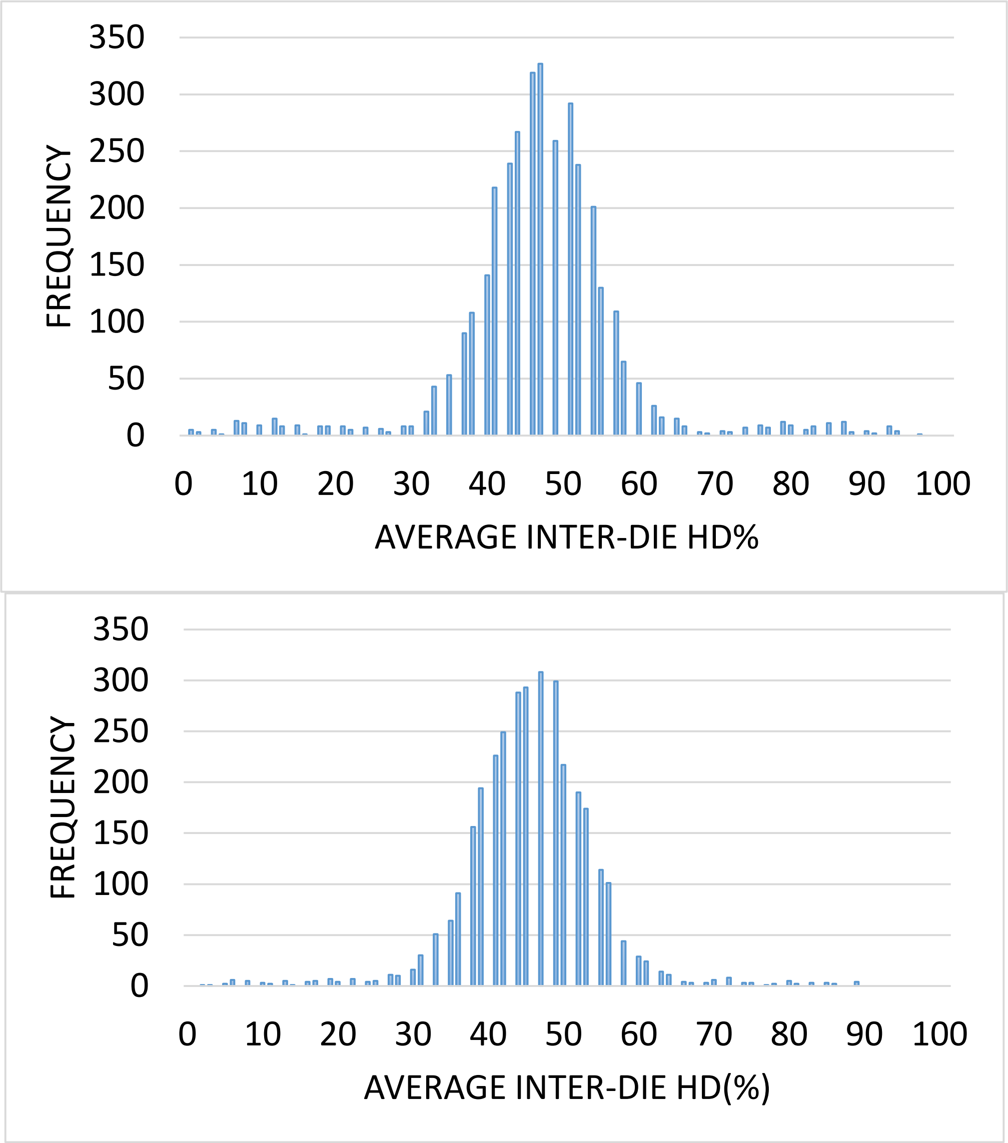


Figure 30. Average Inter-Die Hamming Distance for different challenges

attacks within the additional algorithm and will add overhead. The following equation is used to calculate the uniqueness across k chips of a PUF.

$$Uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} * 100\% \qquad [12]$$

where R_i and R_j represent the response bit streams for challenge i and j respectively and HD is the hamming distance between the two.

Figure 29 and Figure 30 show the distribution of the inter-die hamming distances between the responses of eighty different samples of this PUF for four different randomly chosen input challenges. Table 3. Mean, Standard Deviation and Percent of ones of responses

	Challenge 1	***Challenge 2***	***Challenge 3***	***Challenge 4***	***Total***
Mean	46.69	45.17	48.16	50.08	47.52
Standard Deviation	11.06	8.78	9.06	9.15	9.51
Percent of Ones	54.64	51.99	51.25	48.40	51.57

shows the mean and standard deviation of the four challenges. The overall inter-die hamming distance as indicated in the table is 47.52 percent without any post-processing step, which makes attacks with bias detection algorithms infeasible.

B. Uniformity

Another important metric in characterizing a PUF is its uniformity. It is so named because an ideal PUF will have its bits uniformly distributed between its two states. In effect, uniformity is defined as the average value of the n bit response or otherwise shown by the following equation for the instance i of the PUF circuit:

Table 3. Mean, Standard Deviation and Percent of ones of responses

	Challenge 1	***Challenge 2***	***Challenge 3***	***Challenge 4***	***Total***
Mean	46.69	45.17	48.16	50.08	47.52
Standard Deviation	11.06	8.78	9.06	9.15	9.51
Percent of Ones	54.64	51.99	51.25	48.40	51.57

$$Uniformity = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} * 100\% \qquad [13]$$

Where l represents the bit number and n is the total number of response bits. Table 3. Mean, Standard Deviation and Percent of ones of responses

	Challenge 1	***Challenge 2***	***Challenge 3***	***Challenge 4***	***Total***
Mean	46.69	45.17	48.16	50.08	47.52
Standard Deviation	11.06	8.78	9.06	9.15	9.51
Percent of Ones	54.64	51.99	51.25	48.40	51.57

also represents the uniformity measured of the eighty different instances of the design across four challenges. Overall, a total uniformity of 51.57% has been measured across all 320 responses, which is close to the performance of an idea PUF.

C. Reliability

The last measure for verifying the performance of a PUF is its repeatability. In the case of the proposed PUF, the simulations of the circuit show that the PUF will repeat its performance after every reset since the initial conditions would be the same. However, varying the temperature can affect the response of this PUF. This is mainly because when the variations are introduced on transistors, they will not vary equally with the variation of the temperature. Figure 31 illustrates

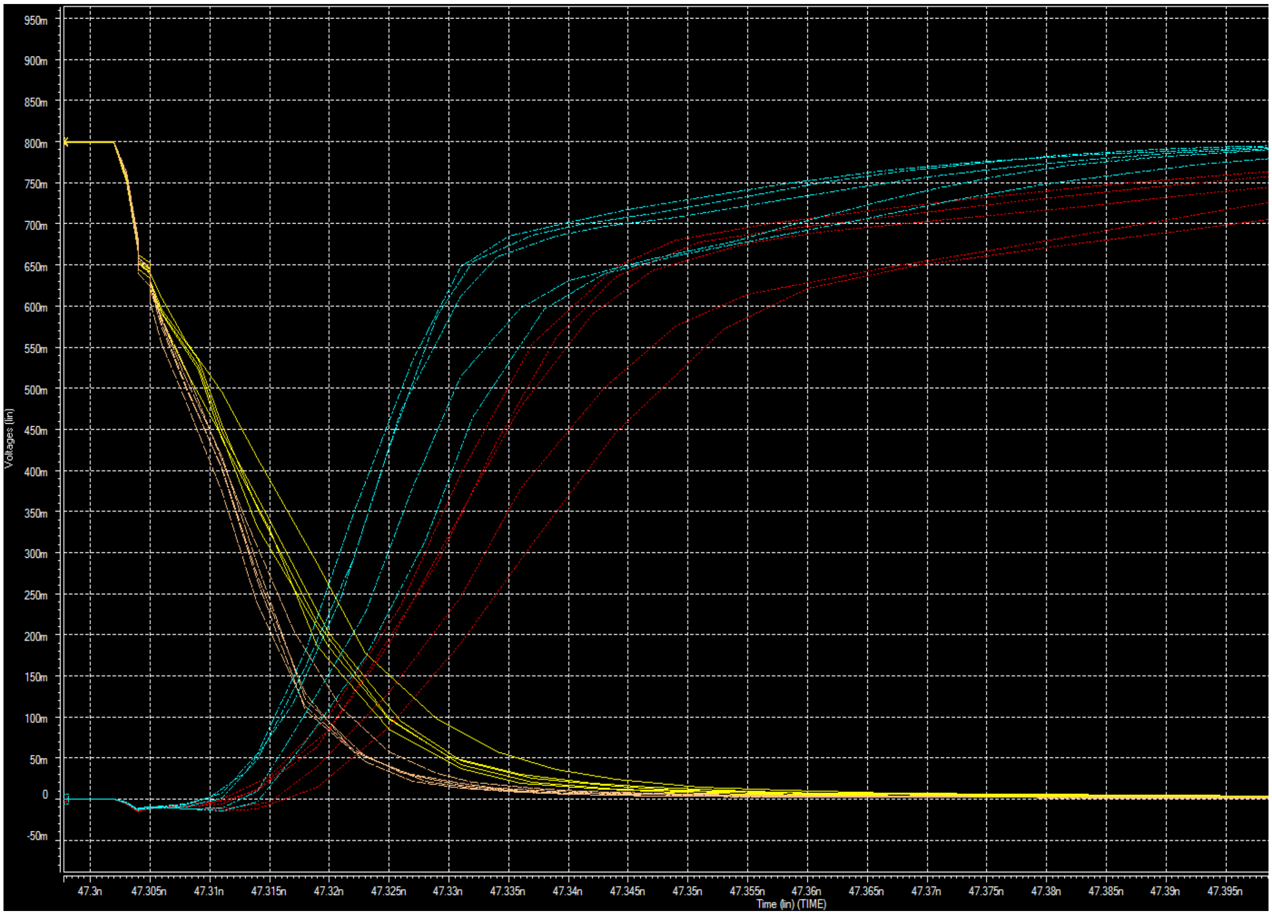


Figure 31. Simulated clock (yellow and bright brown) vs. data (red and blue) for 5 different chips

this scenario. In this figure, blue and red are the voltages after the first switch of the first flip-flop at 0 and 85 degrees centigrade respectively and it is across 5 different chips with global and local variation. In other words, the voltage has been measured for 5 chips once at 0 degrees generating 5 blue curves and another time at 85 degrees across the same 5 chips generating red curves. Alternatively, the clock signal controlling this switch is also shown as brown and yellow colors at 0 and 85 degrees centigrade respectively. As it is illustrated in this figure, given the unknown nature of the process variation the temperature variation will not affect the FF switch and clocks paths equally resulting in this PUF being relatively sensitive to the temperature variation. So it will need to be kept at a relatively constant temperature during characterization and operation phases.

Alternatively, in the environments with high-temperature variations, this PUF can be combined with a temperature sensor to ensure that the chip is at the same temperature that it has been characterized in, before the challenge would be sent. This will guarantee that the response of the PUF would stay close to the expected response and that it will not exceed the accepted error margin.

Even in the absence of a temperature control mechanism, this PUF, can still be utilized in the random number generation applications to generate random numbers. Since these applications do not require a repeatability of the numbers, the proposed PUF will be a simple to implement and low overhead random number generator.

CHAPTER IX: Conclusion

This work exploits the variations in the silicon manufacturing technology and introduces a new delay-based physically unclonable function (PUF). The presented PUF maps these random variations from device parameters such as channel length, width, and threshold voltage to circuit-level random bit streams through violating the setup and hold times of D-flip-flops. These frequent violations are achieved by using the non-multiplicand clock and data frequencies on multiple stages of cascaded D-FFs. Additionally, the random outputs of FFs are further shuffled through the use of XOR gates to avoid characterization attacks. Despite its relative simplicity, this low area overhead PUF generates a vast challenge-response space, which is mostly due to the uncertainties of the outputs of FF while the setup and hold times have been violated. Overall, a 47.52% total hamming distance between the responses of four different challenges with 51.57% uniformity has been measured.

Bibliography

[1] International Technology Roadmap for Semiconductors, http://www.itrs.net/

[2] Andrei Pavolov, Manoj Sachdev. CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies. 1st edition. Springer Publisher, 2008

[3] Yang J, Capodieci L, Sylvester D (2005) Advanced timing analysis based on post-OPC extraction of critical dimensions. In: Proceedings of the IEEE design automation conference, June 2005, pp 359–364

[4] "Analysis and Design of Networks-on-Chip Under High Process Variation" R Ezz-Eldin, MA El-Moursy, HFA Hamed–2016

[5] Blaauw D, Chopra K et al. (2008) Statistical timing analysis: basic principles to state-of-the-art. IEEE Trans Comput Aided Des Integr Circuits Syst 27(4):589–607

[6] Pelgrom MJ, Duinmaijer AC, Welbers PG (1989) Matching properties of MOS transistors. IEEE J Solid-State Circuits 24(5):1433–1439

[7] Wong H-SP, Frank DJ, Solomon PM et al. (1999) Nanoscale CMOS. Proc IEEE 87(4): 537–570

[8] Croon JA, Storms G, Winkelmeier S, Pollentier I (2002) Line-edge roughness: characterization, modeling, and impact on device behavior. In: Proceedings of international electron devices meeting, Dec 2002, pp 307–310

[9] Oldiges P, Qimghuang L, Petrillo K, Leong M, Hargrove M (2000) Modeling line edge roughness effects in sub 100-nanometer gate length devices. In: Proceedings of the international conference on simulation of semiconductor processes and devices, Sept 2000, pp 131–134

[10] Bhaghath P J, Ramesh S R "A Survey of SSTA Techniques with Focus on Accuracy and Speed" International Journal of Computer Applications (0975–8887), Volume 89–No.7, March 2014

[11] S. Walia, "PrimeTime® Advanced OCV Technology," http://www.synopsys.com/Tools/Implementation/SignOff/CapsuleModule/PrimeTime_AdvancedOCV_WP.pdf, April 2009

[12] Yu Zheng; Krishna, A.R.; Bhunia, S., "ScanPUF: Robust ultralow-overhead PUF using scan chain," in Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific , vol., no., pp.626-631, 22-25 Jan. 2013

[13] Tschanz JW, Kao JT, Narendra SG et al. (2002) Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. IEEE J Solid-State Circuits 37(11):1396–1402

[14] Tiwari A, Sarangi SR, Torrellas J (2007) ReCycle: pipeline adaptation to tolerate process variation. In: Proceedings of the international symposium on computer architecture, June 2007, pp 323–334

[15] Ghosh S, Bhunia S, Roy K (2006) A new paradigm for low-power, variation-tolerant circuit synthesis using critical path isolation. In: Proceedings of the intel conference on computer aided design, Nov 2006, pp 619–624

[16] Das, S.; Sanjay Pant; Roberts, D.; Seokwoo Lee; Blaauw, D.; Austin, T.; Mudge, T.; Flautner, K.; , "A self-tuning DVS processor using delay-error detection and correction," *VLSI Circuits, 2005. Digest of Technical Papers. 2005 Symposium on,* vol., no., pp. 258–261, 16-18 June 2005

[17] Yang Liu; Tong Zhang; Parhi, K.K.; , "Analysis of voltage overscaled computer arithmetics in low power signal processing systems," *Signals, Systems and Computers, 2008 42nd Asilomar Conference on* , vol., no., pp.2093-2097, 26-29 Oct. 2008

[18] Dhar, S.; Maksirnovi, D.; Kranzen, B.; , "Closed-loop adaptive voltage scaling controller for

standard-cell ASICs," *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on,* vol., no., pp. 103–107, 2002

[19] Y. Liu, H. Yang, R. P. Dick, H. Wang and L. Shang, "Thermal vs. Energy Optimization for DVFS-Enabled Processors in Embedded Systems," *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on,* San Jose, CA, 2007, pp. 204-209.

[20] Lei Wang; Shanbhag, N. R.; "Energy-efficiency bounds for deep submicron VLSI systems in the presence of noise," *IEEE Trans.on VLSI Systems*, vol.11, no.2, pp. 254–269, Apr. 2003

[21] W. Liu, Renfei; Parhi, K.K.; , "Low-power frequency selective filtering," *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on,* vol., no., pp.245-248, 24-27 May 2009.

[22] Shanbhag, Naresh R.; Abdallah, Rami A.; Kumar, Rakesh; Jones, Douglas L.; , "Stochastic computation," Design Automation Conference (DAC), 2010 47th ACM/IEEE , pp.859-864, 13-18 June 2010

[23] H. Cho, L. Leem and S. Mitra, "ERSA: Error Resilient System Architecture for Probabilistic Applications," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 4, pp. 546-558, April 2012. doi: 10.1109/TCAD.2011.2179038

[24] D. Anderson, J. Dykes, and E. Riedel, \More than an interface-SCSI vs. ATA," in 2nd USENIX Conference on File and Storage Technologies (FAST03), pp. 245–257, 2003.

[25] L. Smarr, \Project greenlight: Optimizing cyber infrastructure for a carbon-constrained world," *Computer*, vol. 43, no. 1, pp. 22{27, 2010.

[26] S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems", IJCAET, 6(4), 440–459, 2014.

[27] WEISER, M., WELCH, B., DEMERS, A. J., AND SHENKER, S. Scheduling for reduced CPU energy. In 1st OSDI (Monterey, CA, USA, Nov 1994), pp. 13–23.

[28] Etienne Le Sueur and Gernot Heiser; ," Dynamic voltage and frequency scaling: The laws of diminishing returns" In 2010 Hot power 2010

[29] Sakurai, T.; Newton, A.R.; , "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *Solid-State Circuits, IEEE Journal of* , vol.25, no.2, pp.584-594, Apr 1990

[30] K. Moiseev, A. Kolodny and S. Wimer. "Timing-aware power-optimal ordering of signals". ACM Transactions on Design Automation of Electronic Systems, Volume 13 Issue 4, September 2008.

[31] Robert B.Hitchcock, Sr, Gordon L. Smith, David D. Cheng, "Timing Analysis of Computer Hardware," IBM Journal, vol. 26, no. 1, Jan 1981.

[32] D. Blaauw, K. Chopra, A. Srivastava and L. Scheffer, "Statistical Timing Analysis: From Basic Principles to State of the Art," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 589-607, April 2008.

[33] I. Nitta, T. Shibuya, K. Homma. "Statistical Static Timing Analysis Technology" FUJITSU Sci. Tech. J., 43,4,p.516-523 (October 2007)

[34] A. Zjajo, Stochastic process variation in Deep-Submicron CMOS: Circuits and Algorithms. United States: 2014.

[35] Orshansky, M.; Keutzer, K., 2002, A general probabilistic framework for worst case timing analysis, Design Automation Conference, 2002. Proceedings. 39th, Vol., Iss., 2002, Pages: 556–561.

[36] Sakurai, T.; Newton, A.R.; , "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," Solid-State Circuits, IEEE Journal of , vol.25, no.2, pp.584-594, Apr 1990

[37] Walia, Sunil; "PrimeTime Advanced OCV Technology "; Synopsys, Inc.; Apr. 2009.

[38] Timing analysis shifts to statistical, Available online: http://www.techdesignforums.com/practice/technique/cadence-socv-statistical-timing-10nm/

[39] M. Weber, "My Head Hurts, My Timing Stinks, and I Don't Love On-chip Variation", SNUG, Boston, 2002.

[40] Incentia Design Systems, Inc, "Advanced On-chip-variation Timing Analysis for Nanometer Designs" Available online at: http://www.incentia.com/products/TimeCraft_01.pdf

[41] Brandon Bautz & Swamy Lokanadham, "A Slew/Load-Dependent Approach to Single-variable Statistical Delay Modeling" Tau Workshop 2014

[42] Samy Zaynoun, Muhammad S Khairy, Ahmed M Eltawil, Fadi J Kurdahi, Amin Khajeh, "Fast error-aware model for arithmetic and logic circuits," Computer Design (ICCD), 2012 IEEE 30th International Conference on., pp.322,328, Sept. 30 2012-Oct. 3 2012.

[43] Jongyoon Jung; Taewhan Kim, "Variation-Aware False Path Analysis Based on Statistical Dynamic Timing Analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol.31, no.11, pp.1684,1697, Nov. 2012

[44] Khajeh, A.; Eltawil, A.M.; Kurdahi, F.J.; , "Embedded Memories Fault-Tolerant Pre- and Post-Silicon Optimization," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.19, no.10, pp.1916-1921, Oct. 2011

[45] Predictive Technology Model (PTM). http://www.eas.asu.edu/~ptm

[46] Y. Liu, T. Zhang, and K. Parhi, "Computation error analysis in digital signal processing system with overscaled supply voltage", IEEE Trans. on VLSI, vol. 18, no. 4, pp. 517-526, Apr. 2010.

[47] R. Maes, "Physically Unclonable Functions" DOI 10.1007/978-3-642-41395-7_2, © Springer-Verlag Berlin Heidelberg 2013

[48] C. Herder, M. D. Yu, F. Koushanfar and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," in *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126-1141, Aug. 2014.

[49] Sergey Morozov, Abhranil Maiti, Patrick Schaumont, "A Comparative Analysis of Delay Based PUF Implementations on FPGA" (2010)

[50] Roel Maes, Ingrid Verbauwhede "Physically unclonable functions: a study on the state of the art and future research directions", Towards Hardware-Intrinsic Security, Security, and Cryptology. (2010)

[51] Stefan Katzenbeisser, et.al "PUFs: Myth, Fact, or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon" CHES'12 Proceedings of the 14th international conference on Cryptographic Hardware and Embedded Systems

[52] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)

[53] V. van der Leest, E. van der Sluis, G. J. Schrijen, P. Tuyls, and H. Handschuh, Efficient implementation of true random number generator based on sram pufs," in Cryptography and Security, 2012, pp. 300-318.

[54] K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in fpga based on oscillator rings," in Reconfigurable Computing and FPGAs, 2008. ReConFig '08. International Conference on, dec. 2008, pp. 385–390.

[55] Tolk, K.: Reflective particle technology for identification of critical components. Tech. Rep. SAND-92-1676C, Sandia National Labs, Albuquerque, NM (1992)

[56] Pappu, R.S.: Physical one-way functions. Ph.D. thesis, Massachusetts Institute of Technology (2001)

[57] Pappu, R.S., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science 297, 2026–2030 (2002)

[58] Buchanan, J.D.R., Cowburn, R.P., Jausovec, A.V., Petit, D., Seem, P., Xiong, G., Atkinson, D., Fenton, K., Allwood, D.A., Bryan, M.T.: Forgery: 'fingerprinting' documents and

packaging. Nature 436(7050), 475 (2005)

[59] Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Cryptographic Hardware and Embedded Systems Workshop, LNCS, vol. 4727, pp. 63–80 (2007)

[60] Mukhopadhyay, S., Mahmoodi, H., Roy, K.: Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS. In: IEEE TCAD, pp. 1859-1880 (2005)

[61] Holcomb, D.E., Burleson, W.P., Fu, K.: Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In: Proceedings of the Conference on RFID Security (2007)

[62] Kumar, S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: Extended abstract: The butterfly PUF protecting IP on every FPGA. In: Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on, pp. 67–70 (2008)

[63] D. Lim, J-W. Lee, B. Gassend, M. Van Dijk, E. Suh and S. Devadas, "Extracting secret keys from integrated circuits," IEEE Trans. VLSI yst.,vol. 13, no. 10, pp. 1200-1205, 2005.

[64] Gassend, B.: Physical Random Functions. Master's thesis, MIT, MA, USA (2003)

[65] A. Maiti and P. Schaumont, "Improved ring oscillator PUF: An FPGAfriendly secure primitive," J. Cryptol., vol. 24, no. 2, pp. 375–397, 2011.

[66] C. Yin and G. Qu, "Temperature-aware cooperative ring oscillator PUF," in Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust, 2009, pp. 36–42.

[67] Majzoobi, M.; Koushanfar, F.; Devadas, S., "FPGA PUF using programmable delay lines," in Information Forensics and Security (WIFS), 2010 IEEE International Workshop on , vol., no., pp.1-6, 12-15 Dec. 2010

[68] Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security, pp. 148–160. ACM Press, New York, NY, USA (2002)

[69] Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Design Automation Conference, pp. 9–14. ACM Press, New York, NY, USA (2007)

[70] Yu Zheng; Krishna, A.R.; Bhunia, S., "ScanPUF: Robust ultralow-overhead PUF using scan chain," in Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific , vol., no., pp.626-631, 22-25 Jan. 2013

[71] Niewenhuis, B.; Blanton, R.D.; Bhargava, M.; Ken Mai, "SCAN-PUF: A low overhead Physically Unclonable Function from scan chain power-up states," in Test Conference (ITC), 2013 IEEE International , vol., no., pp.1-8, 6-13 Sept. 2013

[72] Zheng, Y.; Zhang, F.; Bhunia, S., "DScanPUF: A Delay-Based Physical Unclonable Function Built Into Scan Chain," in Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.PP, no.99, pp.1-1

[73] Karakonstantis, G.; Panagopoulos, G.; Roy, K., "HERQULES: System level cross-layer design exploration for efficient energy-quality trade-offs," *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on* , vol., no., pp.117,122, 18-20 Aug. 2010

[74] Visweswariah, C.; Ravindran, K.; Kalafala, K.; Walker, S.G.; Narayan, S.; Beece, D.K.; Piaget, J.; Venkateswaran, N.; Hemmett, J.G., 2006, First-Order Incremental Block-Based Statistical Timing Analysis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, , Vol.25, Iss.10, Oct. 2006, Pages: 2170–2180

[75] Tschanz JW, Narendra S, Nair R, De V (2003) Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high-performance microprocessors. IEEE J Solid-State Circuits 38(5):826–829

[76] Elgebaly, M.; Sachdev, M.; , "Variation-Aware Adaptive Voltage Scaling System," *Very*

Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol.15, no.5, pp.560-571, May 2007

[77] Gupta P, Heng FL (2004) Toward a systematic-variation aware timing methodology. In: Proceedings of the IEEE design automation conference, July 2004, pp 321–326

[78] Kihwan Choi; Soma, R.; Pedram, M.; , "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.24, no.1, pp. 18–28, Jan. 2005

[79] B. Lin, et. al, User and Process-Driven Dynamic Voltage and Frequency Scaling, in ISPASS, pp. 11-22, 2009.

[80] Yang Liu; Tong Zhang; Parhi, K.K., "Analysis of voltage overscaled computer arithmetics in low power signal processing systems," *Signals, Systems and Computers, 2008 42nd Asilomar Conference on* , vol., no., pp.2093,2097, 26-29 Oct. 2008

[81] Ye, Rong, Ting Wang, Feng Yuan, Rakesh Kumar, and Qiang Xu. "On reconfiguration-oriented approximate adder design and its application." In*Proceedings of the International Conference on Computer-Aided Design*, pp. 48-54. IEEE Press, 2013.

[82] Megalingam, Rajesh Kannan, Gautham Popuri, and Parthasara thy Ravisankar. "Low Power Consumption Coarse-Grained Reconfigurable Adder." In *Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on*, vol. 2, pp. 503-506. IEEE, 2009.

[83] Perri, S., P. Corsonello, and G. Cocorullo. "64-bit reconfigurable adder for low power media processing." *Electronics Letters* 38, no. 9 (2002): 397-399.

[84] Perri, Stefania, Pasquale Corsonello, and Giuseppe Cocorullo. "A high-speed energy-efficient 64-bit reconfigurable binary adder." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 11, no. 5 (2003): 939-943.

Appendix A.

The HSPICE code used (excluding the circuit)

```
.global vcc!
.include "45nm_LP_Customized_vth"

.param vdd=0.8
.param mult = 1
.param l_m =45n
.param w_m ='60n * mult'
.param pnratio=2
.param w_m_p='60n * pnratio * mult'



.param l_m_5d=51n

.param w_m_5d='60n'
.param w_m_p_5d='60n * pnratio'

.param lent=45n
.param wit=45n
.param wit_p='45n*pnratio'
.param varn= 0.26
.param varp= 0.06
.param vthn= 0.426
.param vthp= 0.414

.param var_lent_gl_45n =4.5n
.param var_wit_gl_45n =4.5n
.param var_wit_p_gl_90n =9n

.param var_lent_lo_45n =4.5n
.param var_wit_lo_45n =4.5n
.param var_wit_p_lo_90n =9n

.param lent_gl_c1_double =aunif(lent,var_lent_gl_45n )
.param wit_gl_c1_double =aunif(wit,var_wit_gl_45n)
.param wit_p_gl_c1_double =aunif(wit_p,var_wit_p_gl_90n)

.param lent_gl_c1 =lent_gl_c1_double
.param wit_gl_c1 =wit_gl_c1_double
.param wit_p_gl_c1 =wit_p_gl_c1_double

.param wit_p_gl_c2='wit_p_gl_c1'
.param wit_x_gl='wit_gl_c1 * 6.66'
.param wit_x_p_gl='wit_p_gl_c1 * 11.11'
```

```
.param l_x_m_c1 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_x_m_c1 =aunif(wit_x_gl,45n)
.param l_x_m_p_c1 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_x_m_p_c1 =aunif(wit_x_p_gl,150n)


.param wit_p_gl_c11='wit_p_gl_c1 * 3.88'

.param l_m_c1 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c1 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c1=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c1=aunif(wit_p_gl_c11,52.5n)

.param l_m_c2 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c2 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c2=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c2=aunif(wit_p_gl_c2,var_wit_p_lo_90n)

.param l_m_c3 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c3 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c3=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c3=aunif(wit_p_gl_c2,var_wit_p_lo_90n)

.param l_m_c4 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c4 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c4=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c4=aunif(wit_p_gl_c2,var_wit_p_lo_90n)

.param l_m_c5 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c5 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c5=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c5=aunif(wit_p_gl_c2,var_wit_p_lo_90n)

.param l_m_c6 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c6 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c6=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c6=aunif(wit_p_gl_c2,var_wit_p_lo_90n)

.param l_m_c7 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c7 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c7=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c7=aunif(wit_p_gl_c2,var_wit_p_lo_90n)

.param l_m_c8 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c8 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c8=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c8=aunif(wit_p_gl_c2,var_wit_p_lo_90n)

.param l_m_c9 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c9 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c9=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c9=aunif(wit_p_gl_c2,var_wit_p_lo_90n)
```

```
.param l_m_c0 =aunif(lent_gl_c1,var_lent_lo_45n)
.param l_m_p_c0 =aunif(lent_gl_c1,var_lent_lo_45n)
.param w_m_c0=aunif(wit_gl_c1,var_wit_lo_45n )
.param w_m_p_c0=aunif(wit_p_gl_c2,var_wit_p_lo_90n)


.param varn_c1_glo_double=aunif(0,0.0426)
.param varp_c1_glo_double=aunif(0,0.0414)


.param varn_c1_glo=varn_c1_glo_double
.param varp_c1_glo=varp_c1_glo_double

.param varn2= 0.26
.param varp2= 0.16


.param varn3= 0.036
.param varp3= 0.015


.param  delvton1=agauss(varn_c1_glo, varn , 6)
+       delvtop1=agauss(varp_c1_glo, varp , 6)
.param  delvton3=agauss(varn_c1_glo, varn2 , 6)
+       delvtop3=agauss(varp_c1_glo, varp2 , 6)

.param  delvton2=agauss(varn_c1_glo, varn2 , 6)
+       delvtop2=agauss(varp_c1_glo, varp2 , 6)

.param  delvton4=agauss(varn_c1_glo, varn2 , 6)
+       delvtop4=agauss(varp_c1_glo, varp2 , 6)
.param  delvton5=agauss(varn_c1_glo, varn2 , 6)
+       delvtop5=agauss(varp_c1_glo, varp2 , 6)
.param  delvton6=agauss(varn_c1_glo, varn2 , 6)
+       delvtop6=agauss(varp_c1_glo, varp2 , 6)
.param  delvton7=agauss(varn_c1_glo, varn2 , 6)
+       delvtop7=agauss(varp_c1_glo, varp2 , 6)
.param  delvton8=agauss(varn_c1_glo, varn2 , 6)
+       delvtop8=agauss(varp_c1_glo, varp2 , 6)
.param  delvton9=agauss(varn_c1_glo, varn2 , 6)
+       delvtop9=agauss(varp_c1_glo, varp2 , 6)
.param  delvton0=agauss(varn_c1_glo, varn2 , 6)
+       delvtop0=agauss(varp_c1_glo, varp2 , 6)
.param  delvton10=agauss(varn_c1_glo, varn3 , 6)
+       delvtop10=agauss(varp_c1_glo, varp3 , 6)
.param  delvton11=agauss(varn_c1_glo, varn2 , 6)
+       delvtop11=agauss(varp_c1_glo, varp2 , 6)
.param  delvton12=agauss(varn_c1_glo, varn2 , 6)
+       delvtop12=agauss(varp_c1_glo, varp2 , 6)
```

```
M14983 (net01035 net01051 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14982 (net01039 net01059 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14979 (ckbar4 net01035 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14978 (ck4 net01039 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14974 (net01051 net01055 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14973 (net01055 clk_bar 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14972 (net01059 net01063 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14970 (net01063 clk 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14967 (ck3 net01075 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14966 (ckbar3 net01079 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14963 (net01075 net01087 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14962 (net01079 net01095 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14959 (net01083 clk 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14957 (net01087 net01083 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14956 (net01091 clk_bar 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14955 (net01095 net01091 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14950 (net01099 net01103 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14949 (net01103 clk_bar 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14948 (net01107 net01111 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14946 (net01111 clk 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14943 (net01115 net01099 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14942 (net01119 net01107 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14939 (ckbar2 net01115 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14938 (ck2 net01119 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10
M14935 (ck1 net01139 0 0) nch w=w_m_5d l=l_m_5d   delvto=delvton10

.
.
.
(Description of the rest of the circuit)

.
.
.

V1 (vcc! 0) dc=vdd
V4 (clk_bar 0) pulse vdd 0 0 2p 2p 5n 10n
V3 (clk 0)     pulse 0 vdd 0 2p 2p 5n 10n
V2 (d 0) LFSR (0 vdd 0 2p 2p 570.7784824meg  1 [16,14,13,9] rout=0)
V5 (dbar 0) LFSR (vdd 0 0 2p 2p 570.7784824meg  1 [16,14,13,9] rout=0)

.tran 1n 20.48u sweep monte=1
.temp   0
.option
+          MCBRIEF=1
+      ITRPRT=0
+          NOMOD
+          NOELCK STATF1
+          seed = 19
+      post=2
```

```
.print tran  v(out1)  v(out5) v(out10) v(out15)


.END
```