Error Estimation and Error Reduction with Input-Vector Profiling for Timing Speculation in Digital Circuits

By

Xiaowen Wang

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

May 10, 2019

Nashville, Tennessee

ACKNOWLEDGEMENTS

This is a long journey. I am glad that I finally here after all difficulties and obstacles during these years. For this important achievement in my life, the first and foremost person I would like to thank is my dear advisor, Dr. William H. Robinson, for his continuous guidance and support not only in academic field but also in my life. He always encourage me to follow my curiosity and been there when I need help. I also want to thank all of my committee members, Dr. Bhuva, Dr. Mendenhall, Dr. Loveless, and Dr. Gokhale, for their insightful comments. Thank Vanderbilt University and National Science Foundation for providing all resources and financial support.

Along this journey, I am so grateful to my husband, Zhengyu, for his hearted encouragement during my down time. Without his love and fully support, I am not sure if I have the strength to make to this point.

Lastly, thanks to my parents for always believe in me, you guys are awesome!

To my son, Marvin, and my grandparents, I love you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

With nanometer fabrication technologies, system-on-a-chip (SoC) design enables the potential of billions of transistors to implement a wide range of functionality. But along with the size scaling, transistors are becoming more sensitive to environmental conditions [1], within die variations [2][3], and even input workload variations. Designs reliability has become a greater concern for integrated circuits (ICs) [4][5]. The design corners are analyzed to determine the worst-case delay possibility. Based on these design corners, designers include additional timing margins as a guard band on critical paths. However, using the worst-case design plus guard bands can be very pessimistic, which translates into a loss of performance while executing real applications [6].

Better-Than-Worst-Case Design

To avoid performance loss because of infrequently-occurring, worst-case scenarios, Better-Than-Worst-Case (BTWC) design was introduced to bridge the gap. It is a design style that was first introduced by Bob Colwell, architect of the Intel Pentium Pro and Pentium IV processor. A traditional design methodology sacrifices performance to contain the extreme cases, so as to ensure an error-free design. The essential idea of BTWC design emphasizes operating on average-cases. BTWC design improves performance by allowing certain timing errors to occur during the normal operation, while preserving the correct operation by adding error detection and correction to the design. By controlling the probability of the timing errors to a desired level, a trade-off can be made that results in an overall net performance gain with the error correction penalty.

Figure 1 illustrates a general approach for BTWC design that includes a core computational component coupled with a checker mechanism that validates the semantics of the core's operations [8]. The additional circuitry of a BTWC design will consume extra power and take time to correct any errors. Thus, several features the Checker/Corrector must be considered: (1) small area, (2) power efficiency, and (3) fast correction capability.

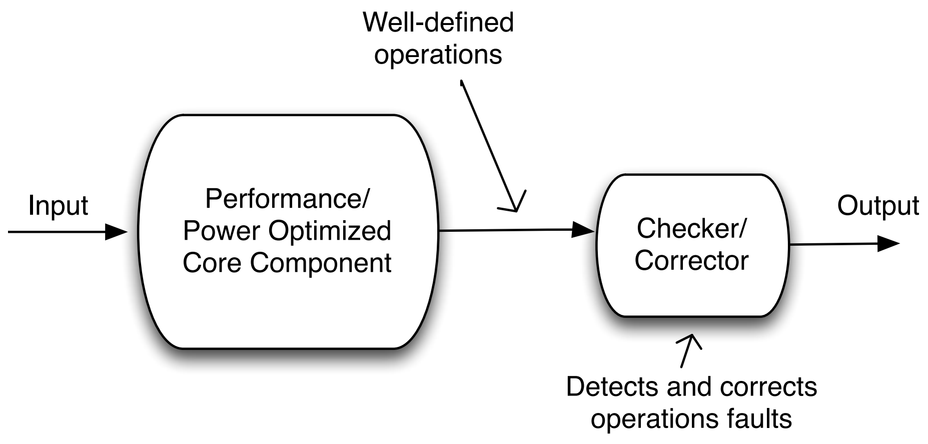

Well-defined operations
Input
Performance/ Power Optimized Core Component
Checker/ Corrector
Output
Detects and corrects operations faults

Figure 1: Block diagram of BTWC design general structure

Figure 2(a) is an example of a critical path delay distribution of a circuit under a certain workload. Figure 2(b) shows the change of the delay distribution of the same path after considering all types of variation. Instead of using clock *l'* as operating clock frequency, BTWC design will select clock *l* to operate at an ultra-high speed, as shown in

Figure 2(b). The gray area represents the error probability. Traditional designs will require a slower clock frequency (i.e., a longer clock period) to avoid errors caused by the probability within the gray area. The advantage of BTWC design is the capability to derive additional performance based upon typical operation cases, and use the error detection and correction (EDAC) circuitry to handle the errors that occur occasionally. However, it will always need a checker/corrector module as long as it contains the possibility of errors. The extra circuitry of the EDAC module will consume extra power and take time to correct any errors. Thus, making the EDAC module small and efficient is important, but selecting a well-balanced operating clock frequency to keep the errors at an ideal level is also the key to maximize performance gain.

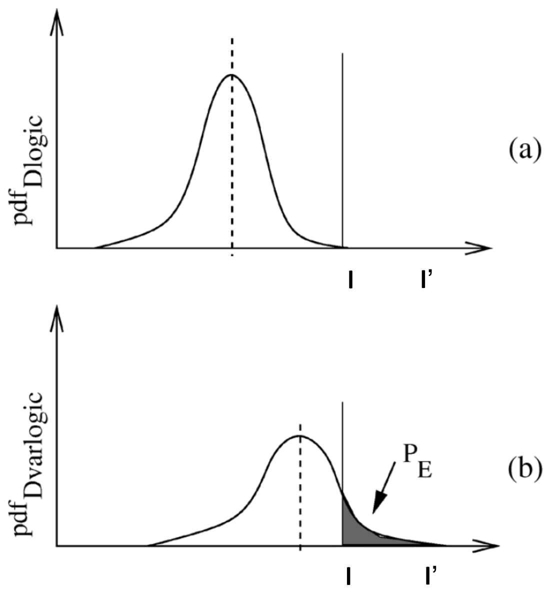


Figure 2: An example of critical path delay distributions: (a) Before the variations and (b) After the variations [7].

The operating clock frequency can be categorized into three regions, as illustrated in Figure 3. **Region One** is the error-free zone, where the clock frequency is usually selected as shown as point ***a***. **Region Two** is where applying timing speculation has a positive performance gain. Timing errors begin to appear beyond point ***b***. Point ***c*** is the optimized clock frequency to maximize performance. **Region Three** is where performance gain becomes negative when applying timing speculation. BTWC designs work within Region Two, therefore identifying the point ***c*** is the ultimate goal of BTWC design.

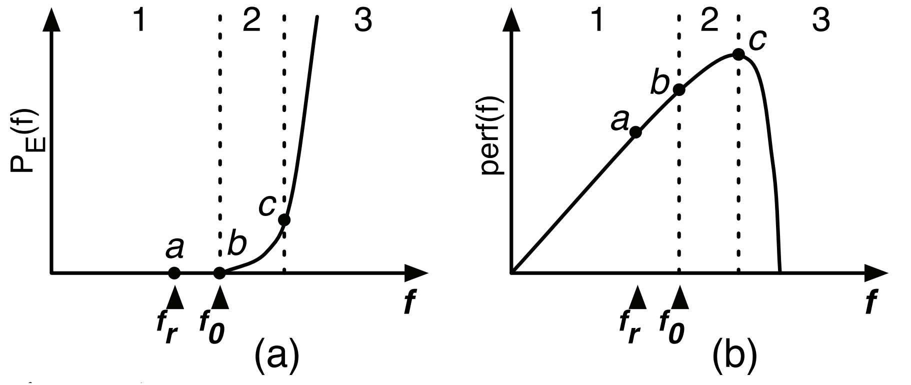


Figure 3: An example to illustrate the relationship of the timing error probability with circuit performance (a) Timing error probability versus clock frequency, (b) Circuit performance versus clock frequency [8].

Variations That Impact Timing

The guard-band is the traditional design approach that tries to contain the timing uncertainties with extra design space. Those timing uncertainties are mostly caused by all kinds of variations, which influences on circuit timing. They can be categorized into static and dynamic sources.

Static variation sources:

Static variation does not change with time and depends on physical factors, such as internal connections and device dimensions. The geometry of the circuit's layout and structure determines the operational parameters. Physical parameter variations (e.g., critical dimension, oxide thickness, channel doping, wire width, wire thickness) lead to electrical parameter variation (e.g., saturation current, gate capacitance, threshold voltage, wire resistance, wire capacitance), and electrical variations result in delay variation. The inability to control precisely the fabrication parameters during manufacturing is called process variation [9]. The partially correlated process variations make the problem complicated to solve. A model of process variation and a model of timing errors for a processor's microarchitecture was described in [7]; it predicts the failure rate of micro-architectural blocks as a function of clock frequency and the amount of variation. A novel approach was proposed in [10] that isolated the failing path to avoid timing errors caused by process variation for fabricated chips.

Dynamic variation sources:

Dynamic variations, on the contrary, are time-related and depend upon the operating conditions, like the fluctuation of: (i) V_{cc} droops, (ii) temperature, (iii) transistor drain current, (iv) cross-coupling capacitance, and (v) multiple inputs switching (MIS) in logic gates [11]. V_{cc} droops are induced from the internal switching activity, and lead to current transients. Dynamic voltage (IR) drop under real switching activity was analyzed in [12]. Temperature depends on input workload, environmental conditions, and heat-control methods. An adaptive system was discussed in [1] that accurately estimates the temperature-induced delay variation to avoid an overly conservative design. Transistor drain current aging is related to the gate bias and temperature. Cross-coupling capacitance change due to the adjacent wires switches will cause RC delay on the wire. MIS is related to input workload that affects the circuits' internal activity and the settling time of the outputs. Paths with the same static propagation delay could have dramatically different distributions of their settling time because of the input workload variation. There are researches like [13] and [14] who have studied the circuit behavior curve under input workload.

Motivation

If the worst static propagation delay of an output is longer than the operating clock period, then there is a probability to observe errors at this output. But a timing error occurs only when the output settling time extends later than the specified clock period. However, when increasing the operating clock frequency, the error probability increases are not linear. Estimating error probability just based on the circuit's static delay information may lead to a severe misunderstanding of the circuit's behavior. The error probability is highly related with the output settling time and the operating clock period. As introduced earlier, there are many factors, including input workload variation, that could influence an output settling time. Most of influences are subtle. It is the input vectors that determine the path usage. Together with circuit's previous status, the basic shape of the circuit dynamic activity curve is formed.

BTWC design is all about finding the optimal operation clock frequency. Timing speculation is associated with errors, but a well-balanced operating clock frequency will contain the error rate at a desired level, so as to realize performance improvement. An accurate error estimation method based on the circuit's dynamic behavior gives BTWC designers insight on how the error trends occur with the increase of clock frequency.

The error rate from each output may vary with different input workload. When a circuit operates at an ultra-high clock frequency, an understanding of the dynamic activity for each circuit path will help to identify the most error-prone output, thereby making the error reduction more efficient. BTWC design requires error estimation for all potential clock frequencies for the given input workload. Therefore, a design flow for BTWC design has been developed to extend the capability of commercial electronic design automation (EDA) tools for dynamic path activity and output settling analysis. The design flow utilizes customized scripts that process standard output files from the commercial EDA tools.

Research Contributions

This research focused on performance improvement in digital integrated circuits (ICs) by considering path activity behavior under a given input workload. Timing analysis and timing closure are critical steps in digital circuit design. Many factors affect the delay distribution of the outputs, but the input workload determines the basic shape of the distribution. The longest static delay path(s) may not be very active for a certain input workload, and therefore would not frequently generate timing errors. Obtaining the actual delay distribution of the outputs for given workload could help designers to estimate the error rate for each output so as to select the well-balanced operating clock frequency, which is the fundamental challenge for BTWC design.

This work contributes the following: (1) The Off-line error checking method that enables detailed statistical analysis on selected cells activities. Design and verifications does not require test bench that compares models operating in system work; (2) The All-clock-frequency error estimation method that predicts error rate for all-possible operation frequency of each PO. Characterizes each PO settling curve under given input workload. Identifies the typical-case error contributor according to circuit internal activity. (3) Dual-V_t method for effective error reduction on selected cells. The selection focuses on the fan-in cone of identified error contributors by using weights determined with circuit activity level. [15].

The rest of this dissertation is organized as follows. Chapter II describes related methodologies and related backgrounds, including: (1) static timing analysis (STA) and statistical static timing analysis methods (SSTA), (2) path stabilization probability analysis for given input workload, and (3) general EDA design flow. Chapter III provides information about the research methodology, simulation setup environment, and the overall design flow used in this work. Chapter IV describes the error-estimation method and the error-checking method used in this research, which aided in identifying the greatest error-contributing primary outputs (POs) and the critical standard library cells that contribute to the propagation delay for the given input workload. Chapter V describes the error reduction method of using multi-threshold standard cells and analyzes the error-reduction results. Chapter VI summarizes the work and offers some future extensions of this work. The customized Python scripts for error checking and error estimation are provided in the Appendix.

CHAPTER II

BACKGROUND AND RELATED WORK

Device reliability is an important concern for the operation of integrated circuits (ICs). Design margins are incorporated into the final design to ensure an error-free IC. BTWC design can be used with timing speculation to reclaim the lost performance when incorporating the design margins associated with corner cases. Timing analysis and path activation probability analysis are necessary to prepare for BTWC design. Understanding the circuit behavior under typical cases is the key to success. The timing error rate of a BTWC design needs to be controlled at an optimized level in order to have an improvement in performance while avoiding the negation of the performance gain because of the error correction penalty.

This chapter provides background on techniques for timing analysis. It also discusses the analysis of path activation probability, which is used to determine how the input workload can affect the visible errors at the primary outputs. The chapter also reviews techniques for timing speculation with error detection and correction (EDAC) circuitry.

Timing Analysis

The original Static-Timing Analysis (STA) was brought into very-large-scale-integration (VLSI) chip design in the early 1990s, and it has been one of the successful and matured tools in digital circuits design [16]. This timing analysis tool could be widely accepted because the runtime is linear to the circuit size, and the results are relatively safe for traditional digital circuit design. The original STA tools are deterministic and calculate circuit delay with one specified corner case to represent the design boundary. However, the transistor size scaling amplified the impact from process variation. The deterministic attribute of traditional STA causes inaccuracies for digital circuits.

The fundamental weakness of traditional STA is that there is no statistically rigorous method for modeling multiple corner files. Therefore, Statistical-Static-Timing-Analysis (SSTA) emerged to improve the timing analysis method. Rather than giving a single result, SSTA evaluates the timing of gates and interconnects with probability distributions. Over the past decade, there are hundreds of papers published in this field, and D. Blaauw et al. [16] discussed the evolution of STA to SSTA. The basic goal of traditional STA is to find the delay of the longest path in the circuit, and the SSTA aims to find the latest arrival-time distribution of the output. The SSTA can be generally categorize into three approaches:

1. ***Numerical-Integration Method:*** The delay distribution of a set of paths that approach the maximum delay can be expressed as a function of physical parameters in order to select a certain region under a specific circuit delay. That region is then integrated numerically to explore the possible physical parameter space, and then compute the circuit's statistical results of the timing. This approach is generic and can include different types of models to account for process variation, but the significant amount of computation time

is a problem, especially for a well-balanced circuit with a large set of paths that approach the maximum delay.

2. ***Monte Carlo simulation method:*** The basic idea behind this approach is to perform sufficient independent sampling for the circuit delay using traditional STA using the probability distribution function (PDF) of the physical parameters. The circuit delay distribution can be found by sweeping the timing constraint. Like numerical integration, this method is completely general. Because the traditional STA methods are mature, Monte Carlo simulation is faster than the numerical integration method. However, due to the inner loop calculation inside of the simulation for STA, the run times are still significant. The second weakness of Monte Carlo simulation method is the difficulty to perform incremental analysis. If any change is made to the circuit, then the whole simulation procedure needs to be restarted to obtain an updated circuit delay distribution.
3. ***Probabilistic analysis method:*** Unlike the previous two methods that are based on the sample space, this method models the gate delay and the arrival time of signals with random variables. There are two main approaches to implement probabilistic analysis.
    a. **Path-based approaches:** This approach selects a set of most-likely critical paths, and then adds the gates and interconnection delays of each path within the set to approximate the circuit delay distribution. The paths selection must be done before the statistical analysis, so the accuracy of the approximation depends on the selection of likely high-

delay paths. This approach, therefore, has two split steps: First, find the paths, and second, calculate the path delay. The difficult task is how to find the best set of paths.

b. **Block-based approaches:** This approach is based on the traditional STA algorithm, and deals with the circuit graph in a topological manner. To compute the arrival time for each node, the edge delay is added with the source node arrival time for each fan-in edge, and then the latest arrival time is selected as the final result for each node. The block-based approach has a runtime advantage, especially because incremental analysis is allowed by using this method.

Although SSTA makes the analysis more comprehensive, there are challenges and limitations associated with it. It is too complex when dealing with realistic delay distributions. It is also very difficult to apply within an optimized algorithm or flow.

Both of traditional STA and SSTA method are designed to avoid the impact from input vectors. It is good for traditional design because including rare cases when testing design limitations makes the design more reliable. However, this advantage becomes an obstacle when using a BTWC design style that optimizes a design according to circuit behavior for typical input workloads.

Path Activation Probability Analysis

Because the rare cases are not emphasized for BTWC design, information from regular timing analysis tools are not enough for BTWC design. Two Primary Outputs (POs) with the same static path delay, according to timing analysis tool, could have dramatically different distributions of path stabilization (i.e., settling time) in a real application [17].

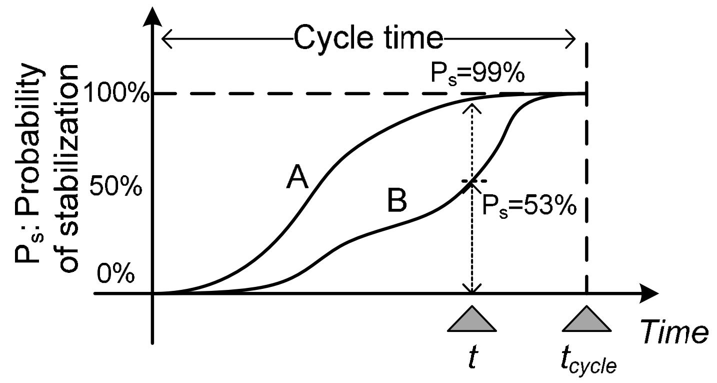


Figure 4: Dynamic behavior curve of two paths with the same static delay time. [17]

Figure 4 shows an example that the path delay time does not equal to the path settling time. The labels A and B represent two outputs that have the same worst-case propagation delay. However Output A and Output B have distinct probability curves for their settling time. With the same input workload, Output A has a 99% probability that it will settle by time *t,* while Output B has a 53% probability to be settled at time *t*. This means Output B is more dynamically critical and should be weighted higher when analyzed for errors during the BTWC design process. By enhancing the speed along the path to only Output B, assuming the circuit only has two paths, then most errors will be reduced, and the circuit could operate at cycle time *t* with very little penalty for error correction.

The circuit's dynamic stabilization curve will be affected by many types of variations, but the curve's basic shape is decided by the input workload. A rigorous analysis of path activation probability, which describes the typical dynamic behavior of the circuit’s response to a common-case workload, would provide insight that helps the BTWC designers to maximize the performance gain. Wan and Chen proposed several circuit optimization techniques for timing speculation based on the circuit’s dynamic activity in [17], [18], [19].

In [18], Wan and Chen proposed a circuit-level optimization tool called DynaTune that combined TCF (Timed Characteristic Function) [20], an ATPG (automatic test pattern generation) method, and BDD (binary decision diagram) [21] to derive the circuit’s dynamic behavior curve to understand the impact of input workload on a circuit’s settling time; based on that information, it selects a targeted operating clock frequency and the corresponding settling probability. Then, it selectively resynthesized the cells along the timing-critical paths that exceed the threshold for delay and activity probability so as to improve performance while mitigating errors. The timing speculation techniques used in DynaTune are the Razor logic [22] (which is discussed in more detail later in the chapter) or the Telescopic Unit [23].

DynaTune has several drawbacks: (1) The use of Global BDD is only suitable for small circuits; (2) TCF analysis is sensitive to the node’s value, and it requires structural information of the circuit to perform the analysis. (3) During the analysis, the input was set to a static probability, which likely is not representative of the real application input workload, which could have distinct phases of operation.

Wan and Chen also purposed a method to analysis circuit-level dynamic behavior with new data structure, called timed Ternary Decision Diagram (tTDD) [17]. The tTDD is created based on the TDD and TCF. Ternary Decision Diagram is similar to Binary Decision Diagrams (BDDs). BDD’s basic idea is Shannon expansion, and it is a graph based rooted but directed data structure that is used to represent Boolean functions. Bryant [24] added restrictions on the ordering of decision variables in vertices, which enables BDD to manipulate representations in a more efficient manner. Ternary decision diagram, As discussed by Sasao in [25], has three possible outgoing branches for each node, which solves BDD incapability of modeling not settled cases. But this method requires circuit partitioning, and the partitioning algorithm is crucial because it will affect both structure correlation and calculation cost. The estimation error complexity becomes relative high when dealing with larger circuits. Detailed timing model that extracted from standard delay format (SDF) was used in this method, but input change impaction on cell delay did not include yet.

CCP [19] resynthesizes a circuit according to a probabilistic manner that creates functionally equivalent but shorter logic paths for paths with high activity. The rarely active paths are resynthesized with a longer delay. To identify the common cases, a global behavior profile is obtained by generating a set of primary input vectors according to given typical case characteristics [8] [26], and then it uses the synthesis engine in ABC [27]. Input vectors are selected according to the typical-case characteristics. The global behavior profile can be reused for all sub-circuits. To promote the common case, the TCF information of common cases is used to build redundant sub-functions for common cases, and the sub-functions are merged into the original design to improve performance.

Timing Speculation Methodologies And Error Resilience in BTWC Design

A BTWC design can be separated into two main parts: Timing speculation part and Error resilience part. Timing speculation is the part to improve the performance (e.g., increase speed, or reduce power usage), which could be implement at different design level. Error resilience part aims to preserve the reliability of the design. In this work, we are focus on the circuit-level timing speculation, and the most popular one is Razor logic structure, like Razor [28], Razor II [29].

Timing speculation methodologies:

Researchers at the University of Michigan developed a circuit-level approach called Razor to implement Dynamic Voltage Scaling (DVS) processors [22][30][29]. It combines a circuit-level error detection mechanism with a microarchitecture-level error recovery technique.

Razor [22] proposed a more aggressive but realistic approach to DVS. It tunes the supply voltage by monitoring the error rate during the operation. The timing error detection is implemented by using a delayed latch, called a shadow latch, to compare with the corresponding state element in the design. The value in the shadow latch is guaranteed to be correct since it uses the worst-case timing (Figure 5). Figure 6 shows how the Razor flip-flop was designed. When an input signal transitions at the same time as the clock, meta-stability may occur in the Razor flip-flop.

Razor relies on both the combinational circuit and the architecture for an efficient EDAC method. It has been applied with a pipeline structure to correct timing errors. Two recovery mechanisms have been proposed [28]. The mechanisms use either clock gating or a counterflow pipeline, as shown in Figure 7 and Figure 8 respectively. The clock gating mechanism simply asserts a global stall for all stages in the next cycle after the error flag is issued. However, global clock gating is not ideal for the clock tree, so the counterflow pipelining approach is introduced. When an error is detected, a bubble signal propagates to next stage, and a pipeline flush is initiated from this stage back to the first stage. The pipeline restarts from the first stage.

The voltage is increased or decreased according to the error rate. A low error rate means that the voltage could be reduced. A high error rate suggests that the supply voltage violates the timing constraints too much and should be increased. A properly selected reference error rate is very important to maximize the performance gain.

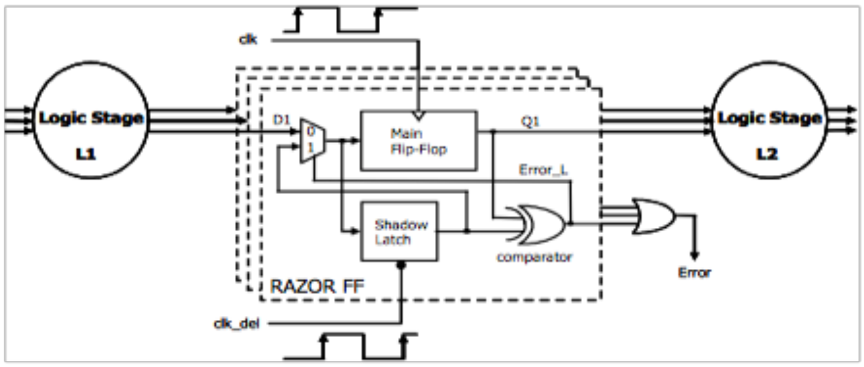


Figure 5: Block diagram of Razor logic. [22]

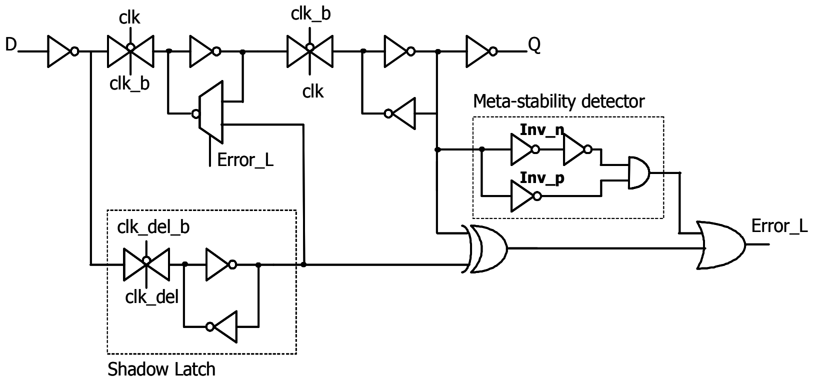


Figure 6: The circuit-level schematic of the shadow latch used in Figure 5. [22]

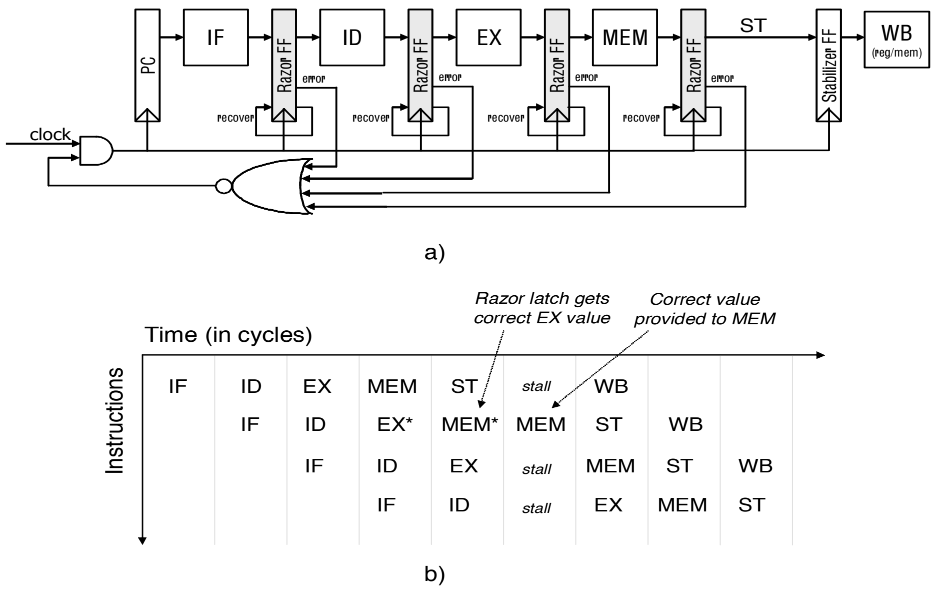


Figure 7: The pipeline recovery using global clock gating. (a) The pipeline structure. (b) The pipeline operation timing. [28]

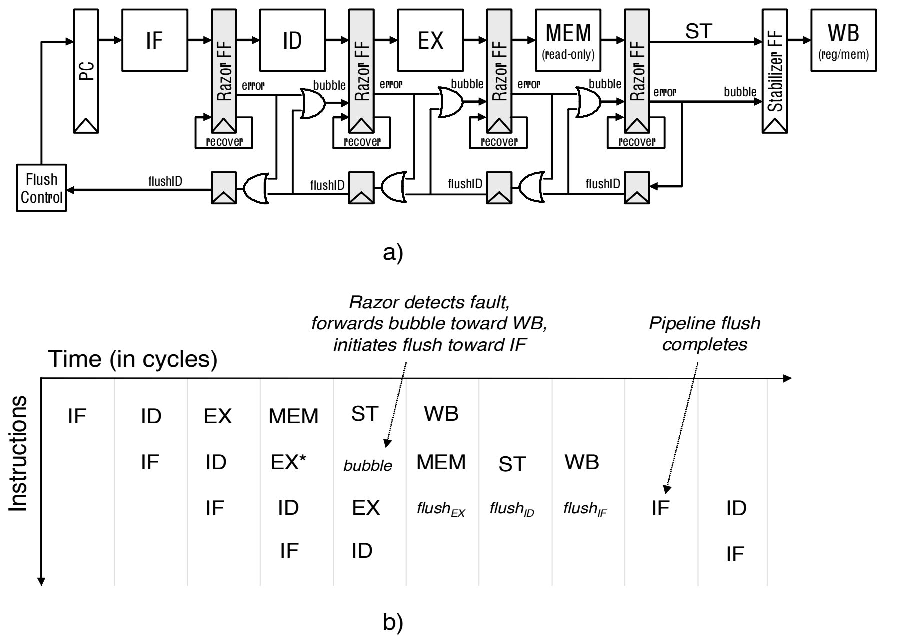


Fig. 8: The pipeline recovery using counterflow pipelining. (a) The pipeline structure. (b) The pipeline operation timing.[28]

The original Razor design not only detects errors but also restores the correct results from the shadow latch. However, generating the restore signal from the pipeline makes it harder to implement an aggressively clocked microprocessor. Razor II [29] proposed a new flip-flop that only detect errors, and uses the technique of architectural replay to handle the correction. Because it uses architectural replay, the Razor II flip-flop is smaller in size and complexity but pays a higher penalty on recovery, as measured by the throughput, Instructions Per Cycle (IPC). The advantage of architectural replay is that it is a mature technique used in many existing speculative processors [31].

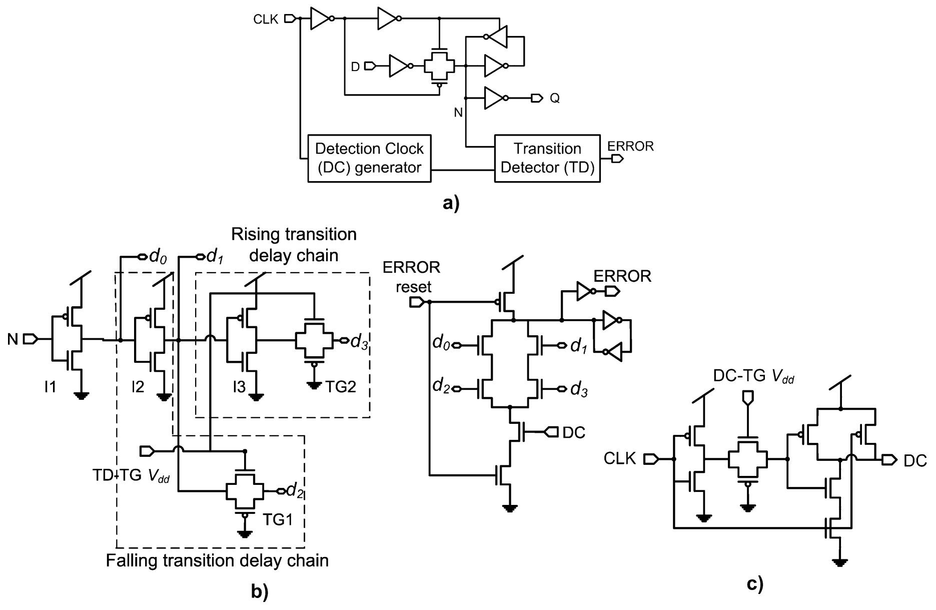


Figure 9: Circuit-level schematic of Razor II flip-flop. (a) Flip-flop schematic. (b) Transition detector schematic. (c) Detection clock generator. [29]

The Razor II flip-flop is a positive level-sensitive latch. Since it is level-sensitive, when the clock is high, any input change will be captured. In Razor II, any transition that happens during the latch's transparent phase is considered an error. Figure 9 illustrates the circuit structure of the Razor II flip-flop.

Other than Razor and Razor II, there are several EDAC flip-flops that use transition detection with time borrowing (TDTB) as in Figure 10(b), and double sampling with timing borrowing (DSTB) as in Figure 10(c). These techniques were proposed to solve the meta-stability issue that exists in the previous Razor design [32]. Figure 10(a) shows the regular structure of Razor flip-flop.

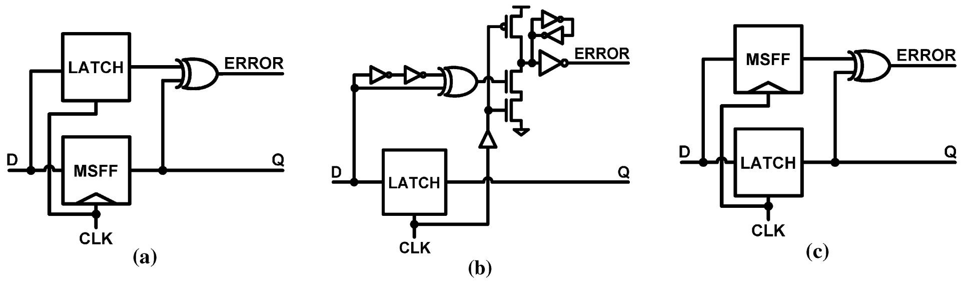


Figure 10: Different ways to implement Razor flip-flop to detect timing errors. [32]

TIMBER [33] proposed two timing elements to provide online masking of timing errors for a pipelined structure. The author found that timing errors due to dynamic variations often only span one pipeline stage on successive clock cycles and therefore can be masked by timing borrowing. TIMBER has flip-flop version and latch version, and both are illustrated in Figure 7 and Figure 8. When **EN** is high, the TIMBER flip-flop works in the timing-borrowing mode. Node **M0** and **M1** are designed for error checking. If they are not the same, then the value sampled by **M1** will mask the previous value after the delay time. The delay time is controlled by $\mathbf{S_1S_0}$. Similar in TIMBER latch, when **EN** is high, the latch is in the timing-borrowing (TB) mode. The input value is latched by transmission gate **M** during the **TB** time interval, while transmission gate **L** is always on for entire checking period in the timing-borrowing mode. If signal arrives after the **TB** interval ends, then the timing error will be masked without an error flag. If signal switching occurred during **ED** interval, then the error flag will be inserted. Both the TIMBER flip-flop and the TIMBER latch do not have meta-stability issue.

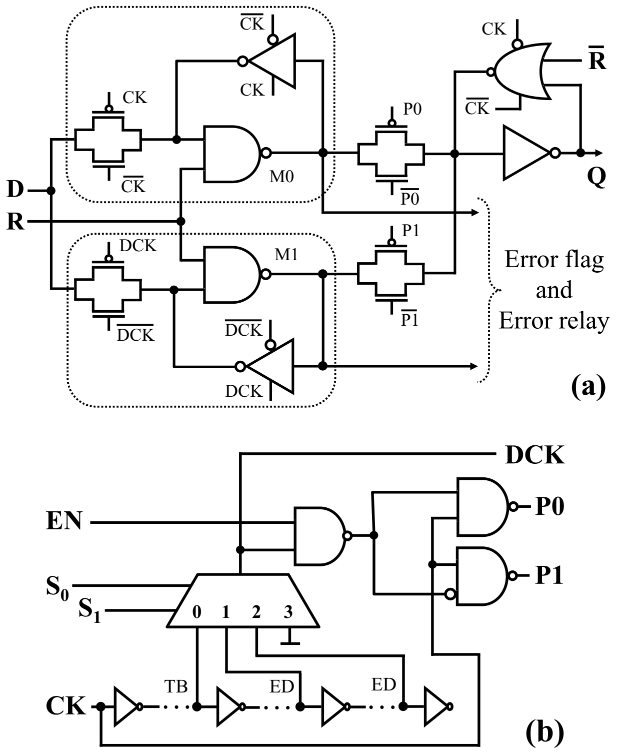


Figure 11: Schematic of TIMBER flip-flop. (a) Main flip-flop part. (b) Clock signal control and generating part.[33]

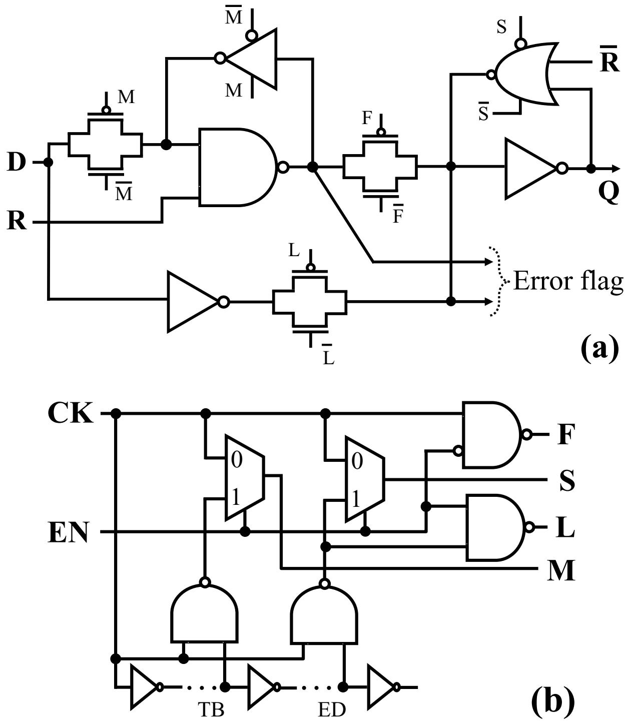


Figure 12: TIMBER latch schematic. (a) Main latch part. (b) Clock signal control and generating part.[33]

TEAtime [34] (Timing error avoidance) uses a methodology that in situ adjusts the clock frequency to avoid operating a circuit at an unnecessarily low frequency. The longest critical path is used as a checker for the main circuitry to ensure correct operation, shown in Figure 13. A toggle flip-flop feeds into the checker to test whether the results could propagate beyond the longest delay under the current clock period. When the checker remain equal, the counter increments, the voltage increases, and the clock frequency increases. The clock frequency can be decreased by implementing the process in reverse. A bi-directional counter, a digital-to-analog (D/A) converter, and a voltage-controlled oscillator (VCO) are used in TEAtime. The prototype design can experience meta-stability.

Error resilience mechanisms:

Error resilience is another part of BTWC designs. The mechanism of each methodology could be categorized as follow:

1. ***Error detection + Rollback/Instruction replay*****:** Normally the approaches that use this scheme include duplicated registers or a transition detection mechanism with a delayed clock to capture signals that violate timing. To recover from the timing errors, the main system is suspended and restored to the correct value from either the duplicated register or a replay of the instruction. Pitfalls of this scheme normally are: (1) a limited detection window, (2) a prolonged hold time requirement, and (3) the issue of meta-stability.
2. ***Error masking*****:** For a given logic circuit, errors can be masked by an approximate logic circuit that predicts the correct value [35], [36]. For every output, the logic could be either expressed with a 0-implication or a 1-implication approximate function. These functions are used to detect 1-to-0 or 0-to-1 errors. The type of approximate function for the output is determined by computing the dominant type of errors.

The work for this dissertation first needed to identify the optimized operating clock frequency (with the assumption of a known threshold value for maximum error tolerance). Then, the design is modified to reduce errors according to the typical activity for the given input workload. The traditional EDAC modules cannot provide an estimation of the error rate for a speculative operating clock frequency unless the simulation is actually performed. However, conducting simulation sweeps through all-possible clock frequencies is too inefficient to accept in a real-world design flow. Therefore, an all-clock-frequency error estimation method has been developed as part of this dissertation research, which enables an accurate error prediction for all-possible speculative operating clock frequencies of each primary output with only one simulation at the original, error-free operating frequency.

Table 1: Summary of several EDAC methodologies

	Detection Methods	**EDAC type**	**Recovery Method**	**Application Structure**	**Pros and Cons**
Razor	Shadow latch	Detection	Restore from register	Pipeline	Meta-stability; Complexity
Razor II	Transition detection	Detection	Architecturally handle instruction replay	Pipeline	Complexity
Bubble Razor	Shadow latch	Detection	Local replay	Pipeline	Less hold time to restore
TIMBER	Duplicate paths	Partial Error detection; Partial Error masking	No	Standard sequential circuit	Limited functions
TEAtime	Monitor critical path	Error masking	Instruction replay	Standard sequential circuit	One path monitor; Meta-stability

<u>Evaluation Methods For BTWC Design</u>

Circuit's performance can be evaluated from three aspects: (1) operational speed, (2) power consumption, and (3) operational reliability. BTWC designs attempt to either enhance computational efficiency or lower the energy usage while maintaining a robust design.

Performance:

The metric for operational speed can use the clock frequency (f), but normally the throughput is used to measure the performance of a BTWC design. The input may need to stall several cycles for the correction penalty when a timing error occurs in a BTWC design. For a traditional design, all the primary outputs (POs) are bounded with the desired cycle time (T_{cycle}), so the probability (P_s) of each PO to stabilize (i.e., settle) within the cycle time is 1. Therefore, assuming that one operation is completed per cycle, the throughput for the traditional design should be [19]:

$$Throughput = P_s \cdot f = f = 1/T_{cycle}$$

However, idea of BTWC design is to make the highly active paths with a long delay to settle before the cycle time, while permitting some of the less time-critical paths to exceed the boundary on occasion. Thus, considering the error correction penalty (*r*), the equation to calculate BTWC design throughput is [19]:

$$Throughput = P_s \cdot f + (1 - P_s) \cdot \frac{f}{r},$$

and the energy cost of BTWC designs is inversely proportional to the throughput.

Power:

The power consumption in the conventional CMOS digital circuit can be separated into three types of dissipation [37][38][39][40]: (1) switching power, (2) short-circuit power, and (3) leakage power consumption. The switching power represents the power dissipated during the signal transitions when energy is drawn from the power supply to charge-up the device capacitances. Short-circuit power is produced during the moment that both the PMOS network and the NMOS network are simultaneously on in CMOS logic. The MOSFETs in CMOS logic normally will have some non-zero reverse leakage and sub-threshold current, which causes the leakage power dissipation. The sum of switching power and short-circuit power can be categorized as dynamic power, while the leakage power is also called static power dissipation [41]. Dynamic power is dominated by switching power, while leakage mainly comes from the sub-threshold leakage current. The static power increases faster than dynamic power with the shrinking of feature size. Reducing supply voltage is an efficient way to reduce total power consumption, but it may lead to timing delay and exponential leakage increase [29][31]. Multi-threshold voltage, where a low-threshold voltage is used with cells on critical paths and a high-threshold voltage is used for the other cells, is a widely accepted technique to reduce power [43]. It has also been used in BTWC design to improve power.

Reliability:

The reliability in BTWC design focuses on the detection and correction of timing errors. Because of the nature of BTWC design, timing errors would invalidate the results during the operation. An effective error detection and correction (EDAC) mechanism is crucial. The evaluation criteria includes answering the following questions:

- How complex is the implementation of the method?
- What penalty is the design going to pay?
- What is the detection/correction rate for the method?

Timing speculation is the idea where various methodologies are used to enhance the operational speed to the point where timing errors occur while equipping the design with techniques to detect and correct those timing errors [44][8]. Based on this idea, BTWC design allows the timing error rate to a certain point where the performance gain (either in speed or in power) is effectively balanced with the penalty cost for reliability. BTWC design has adopted a cross-layered approach [45][46][47] from the architectural level down to circuit level.

Multi-threshold Technology In VLSI Designs

Threshold voltage is the minimum voltage applied on a MOSFET gate to create a conducting path between the source and the drain. The MOSFET acts like a switch ideally. However, during the OFF stage, there are mobile carriers (i.e., electrons or holes) that travel through the semiconductor junctions, which is called sub-threshold leakage. With the technology scaling, the leakage power consumption is now a major concern for current semiconductor industry, and the sub-threshold leakage is the main contributor to the leakage current.

The sub-threshold leakage is directly related to the threshold voltage as it controls the size of the depletion region. A higher threshold voltage could reduce the sub-threshold leakage, but it limits the cell's response speed. On the other hand, a lower threshold voltage reduces the propagation delay but will result in a dramatic increase of leakage power with such small geometry devices.

Many previous works studied how to use multiple threshold voltages on one design. Mutoh et al. [48] introduced the multi-threshold technology for 0.5-μm CMOS that uses low-threshold MOSFETs to enhance speed while high-threshold MOSFETs are used to reduce leakage power. Wang and Vrudhula [49] introduced a heuristic algorithm based on circuit graph enumeration to effectively reduce leakage power of CMOS digital circuit without too much impact on speed. Wei et al. [50] proposed a dual threshold approach to reduce leakage power by assigning high-threshold voltage cells to non-critical paths, and using low-threshold voltage cells on critical paths, and introduced an algorithm to optimize the selection.

The tradeoffs for dual V_t CMOS circuits has been has been explored by Wang and Vrudhula in [51]. The detailed simulation has performed to investigate short circuit power dissipation of dual V_t technology, and the short current impact of low-V_t MOSFETs on gate delay. Multiple power models of dual V_t technology create challenges to EDA tool development as well.

Jayakuamr and Khatri [52] prepared pull-up circuit and a pull-down circuit with different V_t standard cells for standby mode. After the traditional mapping using regular cells, they then replace the cells with prepared low-leakage cells according to the simulation results of each gate's output. The methodology is compared with regular multi-threshold CMOS methodology and shows better performance on leakage reduction.

Most of previous Dual-V_t /Multi-V_t methodologies were targeted to reduce power consumption or to ensure resiliency when applying dynamic voltage scaling (DVS). In this research, the Dual-V_t technology will be used to adjust the timing of specific paths to precisely reduce timing errors during timing speculation.

Timing Speculation vs. Instruction Speculation

Speculation could have different interpretations for people from different research fields. One ambiguity comes from the computer architecture field where researchers commonly refer to the speculative execution based on the branch prediction or out-of-order executions. Based on the history of branch executions, the speculative execution schemes allow the instructions to be scheduled ahead when the outcome of a conditional branch has not yet been determined, in order to utilize the microarchitectural resources in a more efficiently way [31]. However, this widely used optimization technique in modern computer architecture shows security vulnerabilities in January 2018, which affects Intel x86 microprocessors, IBM POWER processors, and some ARM-based microprocessors. One of the vulnerability, Meltdown [53], occurs between memory accesses and privilege checking during instruction processing. The microprocessor's cache holds the unauthorized address because of the out-of-order execution, from which the data can be recovered. The other vulnerability, Spectre [54], uses the information leakage from branch predictions via cache timing as a side-channel attack to manipulate the target process.

The timing speculation discussed in this dissertation is approached from the circuit-level. The traditional clock frequency is bounded by the worst-case delays. Operating the circuit at a higher clock frequency to gain execution speed is the purpose of circuit-level timing speculation. There is no structural modification to the circuit, or out-of-order instruction manipulation in this work to achieve circuit-level timing speculation. Therefore, it does not enable the side-channel attack that has been used for Spectre and Meltdown.

CHAPTER III

RESEARCH METHODOLOGY AND BTWC DESIGN FLOW

This chapter introduces the details of the methodology used for the research in this dissertation. The chapter describes the simulation environment setup as well as the BTWC design flows used to evaluate the approach. Four benchmark circuits from ISCAS85 [48][49] were used to represent four different types of functions (Table 2).

Table 2: Overview of the circuits used in the analysis

Name	Function	Input #	Output #	Cell #
C432	27-channel interrupt controller	36	7	160
C880	8-bits ALU	60	26	383
C1908	16-bit SEC/DED	33	25	880
C6288	16x16-bit multiplier)	32	32	2406

General EDA Flow And The Design Flow Used In This Work

The General EDA flow with Synopsys tools [57] is shown in Figure 13 to give a brief introduction to using the Synopsys design tool kit. Circuit functionality is verified during RTL simulation. Design Compiler (DC) synthesizes the benchmark source file with standard cell modules to generate a gate-level netlist. IC compiler (ICC) performs Place-and-Route, which will generate accurate timing information and store the information in the .sdf file. The post-simulations are performed in VCS with back-annotated timing information (.sdf). The .vcd file stores the simulation results with all the switching activity information for the circuit across the entire simulation time; this file is the essential raw data for the research.

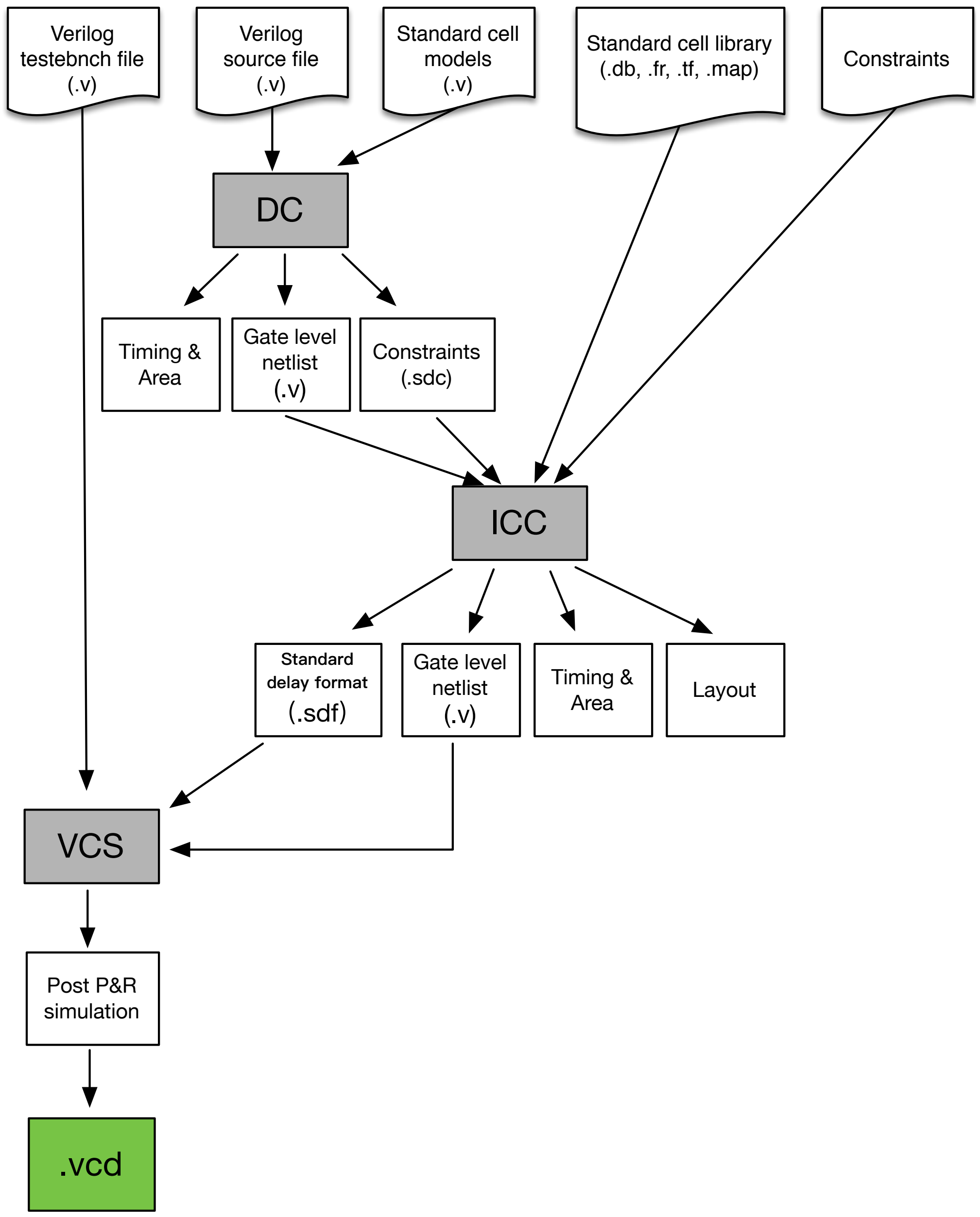


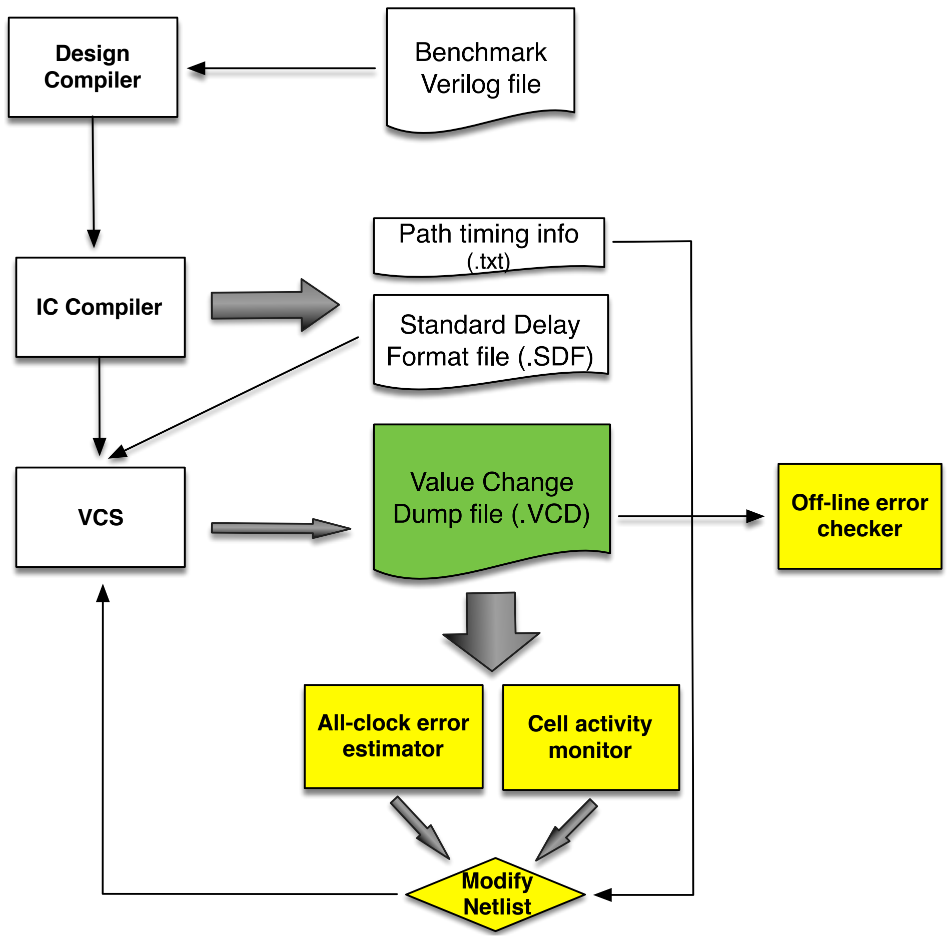
Figure 13: The customized EDA flow with Synopsys tools.

In Figure 14, the customized design flow used in this research is shown. The whole design flow uses the Synopsys Design tool suite (Design Compiler, IC Compiler, PrimeTime, and VCS) with the Synopsys 32-nm library and customized Python scripts to implement, simulate, and analyze the designs.

- After implementation with Design Compiler, the test circuit is place and routed by IC Compiler. The whole circuit delay information, including wire interconnection delay, is saved in the standard delay format file (.sdf file). The gate-level net-list from IC Compiler is simulated in VCS with timing information .sdf file back-annotated. After simulation, the value change dump file (.vcd file) is generated, which records all nodes' switching activity during the simulation time. For the same given input workload of a test circuit, each possible operating clock frequency will have a .vcd file.
- There are several customized scripts to process the .vcd files, which perform the following tasks: (1) Process the .vcd files to prepare and extract the important information for later use; (2) Produce error estimation for a specific speculative clock frequency based on the statistical histogram of each PO's settling time to identify the real error-contributing PO for the given workload; and (3) Calculate the desired internal cells activity level to identify the critical cells.
- According to the cell replacement rule, re-synthesize the test circuit with low threshold voltage (Low-V_t) cells on identified critical cells.
- After the Dual-V_t resynthesis method, use the proposed error-checking method to test the real error rate. The error-checking method uses the same activity .txt file processed from .vcd files as error-estimation. Compare the POs' settling status of the

speculative clock frequency's .vcd file with the golden copy's .vcd file cycle by cycle, and calculate the error count of the POs.

The details of each step are discussed in more detail in the following Chapters.

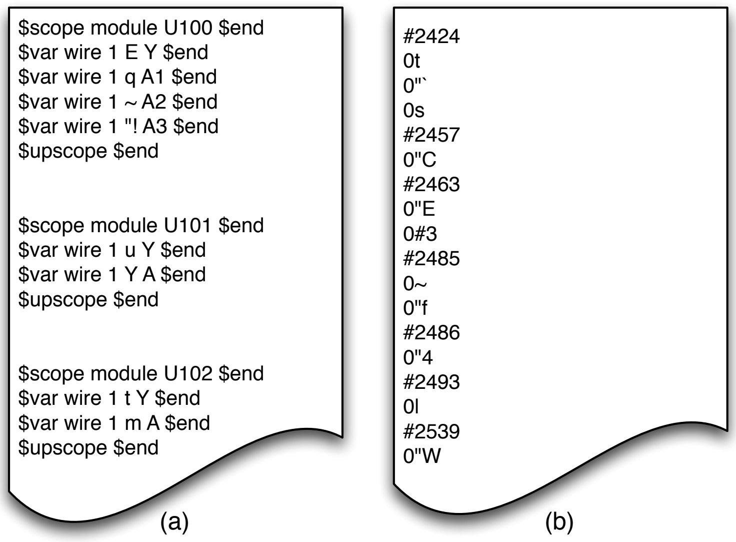


•

Figure 14: The proposed design flow chart.

Value Change Dump Files

The .vcd file was a significant component in this work. It was used not only for the error checking method, but it was also the basis for error estimation and other activity analysis in this work. The .vcd file contains both the switching activity and the timing information that is generated during the simulation. The .vcd file used in this work is generated from the Synopsys simulator. Figure 15 is an example of a .vcd file from an actual simulation. A .vcd file records the switching activity of all the nodes or a selected hierarchy. Figure 15(a) is the header file that defines the corresponding relationship between the node name and the symbol that is used later in the .vcd file. Each node has an assigned symbol. Figure 15(b) records the value change activity of all nodes throughout the simulation time. The entries starting with the symbol "#" are timestamps. The subsequent entries are the nodes that switched at the time point. The first digit is the current value of the node, and the remaining digits are the corresponding symbol of the node. The time unit is ps. One node may change multiple times within one cycle.



```
$scope module U100 $end
$var wire 1 E Y $end
$var wire 1 q A1 $end
$var wire 1 ~ A2 $end
$var wire 1 "! A3 $end
$upscope $end


$scope module U101 $end
$var wire 1 u Y $end
$var wire 1 Y A $end
$upscope $end


$scope module U102 $end
$var wire 1 t Y $end
$var wire 1 m A $end
$upscope $end
```

(a)

```
#2424
0t
0"`
0s
#2457
0"C
#2463
0"E
0#3
#2485
0~
0"f
#2486
0"4
#2493
0l
#2539
0"W
```

(b)

Figure 15: An example of value change dump file. (a) is the header part, and (b) is the body part.

With the understanding of the .vcd file contents, the special customized error estimator and error checker in this work are all based on the information contained in

.vcd file. More details about error estimation and error checking methods will be introduced in Chapter IV and Chapter V.

CHAPTER IV

ALL-CLOCK-FREQUENCY ERROR-ESTIMATION

In traditional design, the designer normally focuses on the task of reducing propagation delay of the static critical paths in order to improve the operating speed of the circuit. The timing errors are observed at the PO when: (1) the PO is activated by an input vector during that cycle, and (2) the settling time is longer than the current operating clock frequency. The PO corresponding to the static critical path may not always lead to the largest error-contributor. The identification of the real error-contributors will help to reduce errors more effectively. However, no existing commercial EDA tool directly provided internal activity analysis coupled with error estimation information. In this chapter, the detailed methodology of the all-clock-frequency error-estimation is described, and error estimation results are discussed.

Obtaining Outputs Settling Behavior

As shown in Chapter III, the .vcd file contains all nodes switching activity and switching time stamps. A timing error occurs when an output has settled after the required clock time. For each PO, only the last switching time of each cycle is important for error estimation. By processing and analyzing the switching information of the output nodes saved in the .vcd file, it is possible to characterize the settling behavior of the primary outputs.

The error estimation methodology in this work is designed to analyze each PO individually. To realize the error estimation methodology, the first step is to extract all of the switching timestamps of the selected PO node. Each cycle has an entry. Then, the last switching time stamp is recorded for this PO at every cycle. A histogram can then be plotted to obtain this PO's settling behavior for the given input workload. The histogram helps to predict the real error contributing POs for the given workload. Figure 16 is the flow chart of the algorithm to take the raw .vcd file and process it in order to extract the timestamps and node activity. The example codes of Benchmark C1908, PO N892 are listed in Appendix Part A. In Part A Section (1), the code prepares all transition timestamps of the given PO, and in Part A Section (2), the code extracts the settling timestamps of the given PO. The bash scripts to automate the process were not included in the Appendix.

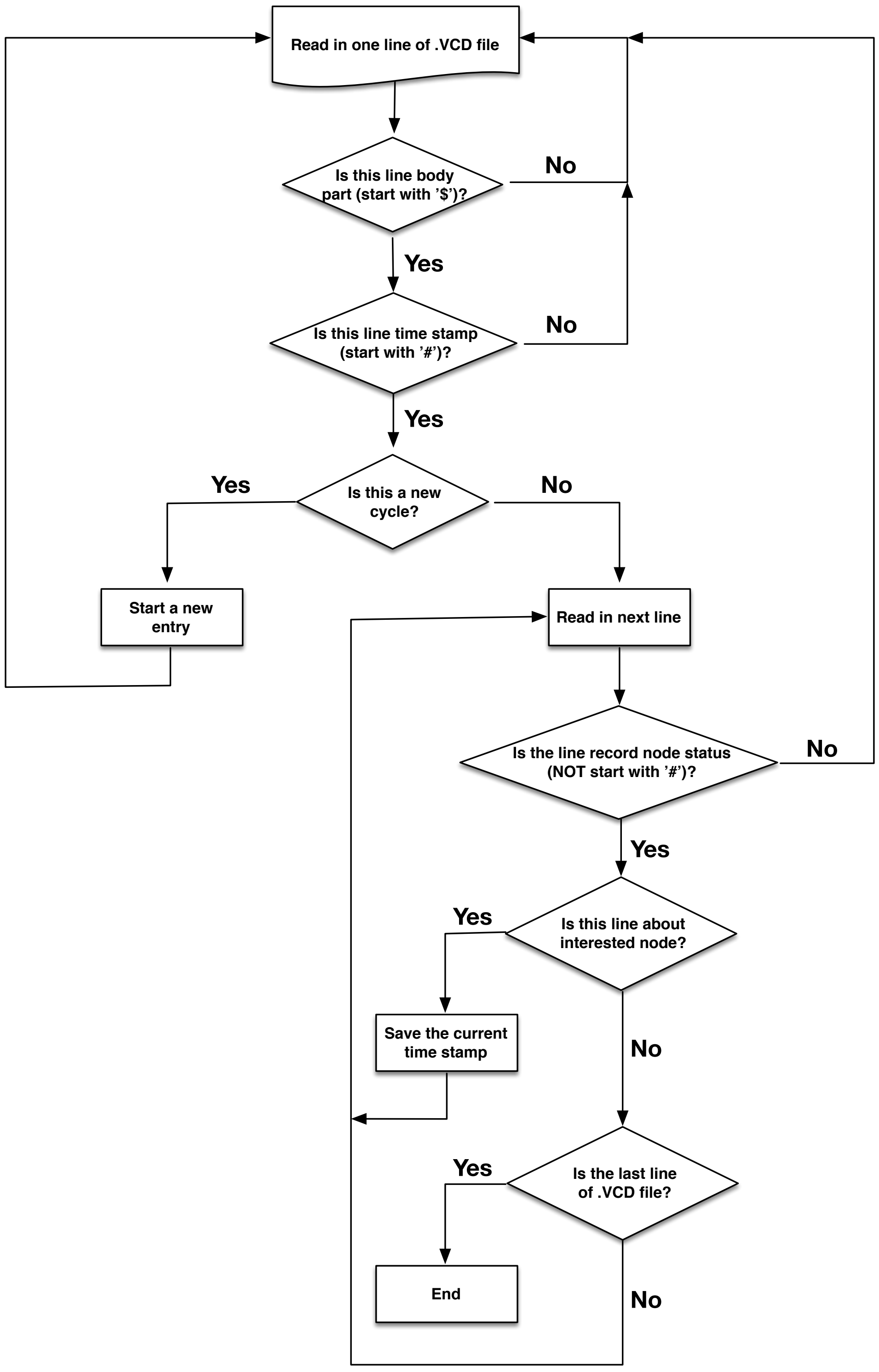

Read in one line of .VCD file
Is this line body part (start with '$')?
No
Yes
Is this line time stamp (start with '#')?
No
Yes
Is this a new cycle?
Yes
No
Start a new entry
Read in next line
Is the line record node status (NOT start with '#')?
No
Yes
Is this line about interested node?
Yes
Save the current time stamp
No
Is the last line of .VCD file?
Yes
End
No

Figure 16: Algorithm flow chart of switching time stamp extraction for a specific node.

Categories Of Primary Outputs

The operating speed of a circuit has traditionally been based on the longest static propagation delay. However, this path may not always be the most active one. This discrepancy can lead to faster, highly active paths that produce more observable errors. Commercial EDA tools do not directly provide the internal activity analysis for the various paths of the circuit. Knowing how frequent a PO will be active by the given input workload, and how likely the PO settles later than the required clock time are the key factors to identify the real error-contributor POs. Then, the information can be used to optimize for BTWC design.

In a circuit, the POs can be categorized into several types:

I. **Safe-POs:** all paths to the output have a shorter propagation delay than the clock period.

II. **Error-possible-POs:** The worst case to the output has a longer propagation delay than the clock period.

III. **Error-prone-POs:** The worst case to the output that has a longer propagation delay than the clock period and has high switching activity.

Category II and Category III have overlap when only considering the PO's activity level, because activity does not directly indicate the error rate. In fact, using just the output's activity level to predict the real error-contributors is not accurate enough.

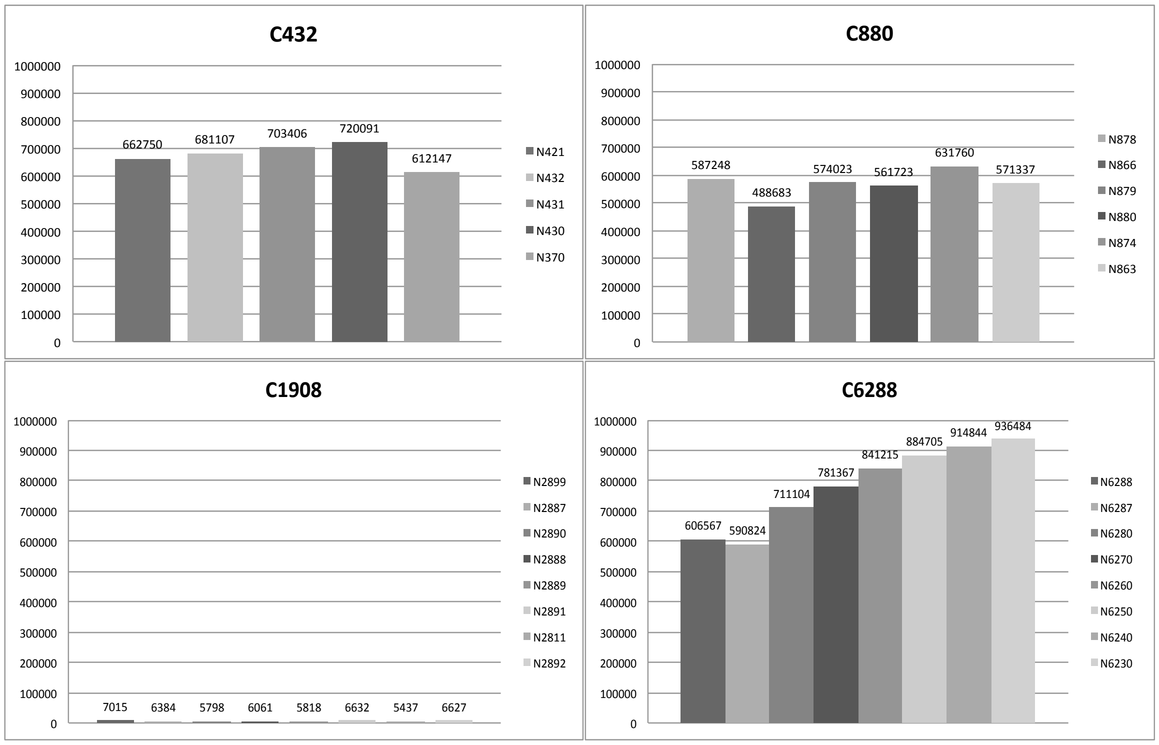


Figure 17: The total active cycles out of 1 million cycles of all error-possible outputs. Benchmark circuits (a) C432, (b) C880, (c) C1908 and (d) C6288. Y-axis shows active cycles with maximum limit 1,000,000 cycles.

Figure 17 shows the total active time of each output, which reflects the activity level of each output under a given input workload. An output could change multiple times during a cycle, and the active cycle count is increased by exactly one when there is one or multiple switches within a cycle. The most active output may not be the largest error contributor during the timing speculation.

For benchmark C1908 as an example in Table 3, PO N2891 is the greatest error-contributor, however neither the switching activity rate nor the active cycles rate indicates the trends. Figure 18 shows the activity level and the actual error count of each output of Benchmark C1908 at the clock period of 1.7 ns. Note that the original clock period is 2.1 ns. The outputs are listed in increasing order of static propagation delay from smallest to largest. Output N2899 has the longest static delay, and the N2886 is the most active one. However, N2891 is the largest error-contributor.

Table 3: Comparison of C1908 static delay, switching activity rate and active cycles rate. Output N2899 has longest delay, while Output N2891 is the greatest error-contributor.

	Static Delay (ns)	**Total Switching Activity Rate**	**Total Active Cycles Rate**	**Stabilization probability at CLK=1.7 ns**
N2899	2.20	1.02%	0.7%	100%
N2887	2.18	0.86%	0.64%	100%
N2890	2.18	0.71%	0.58%	100%
N2888	2.17	0.76%	0.61%	100%
N2889	2.17	0.71%	0.58%	100%
N2891	2.07	0.92%	0.66%	99.9987%
N2811	1.97	0.63%	0.54%	99.9996%
N2892	1.81	0.95%	0.66%	99.9999%

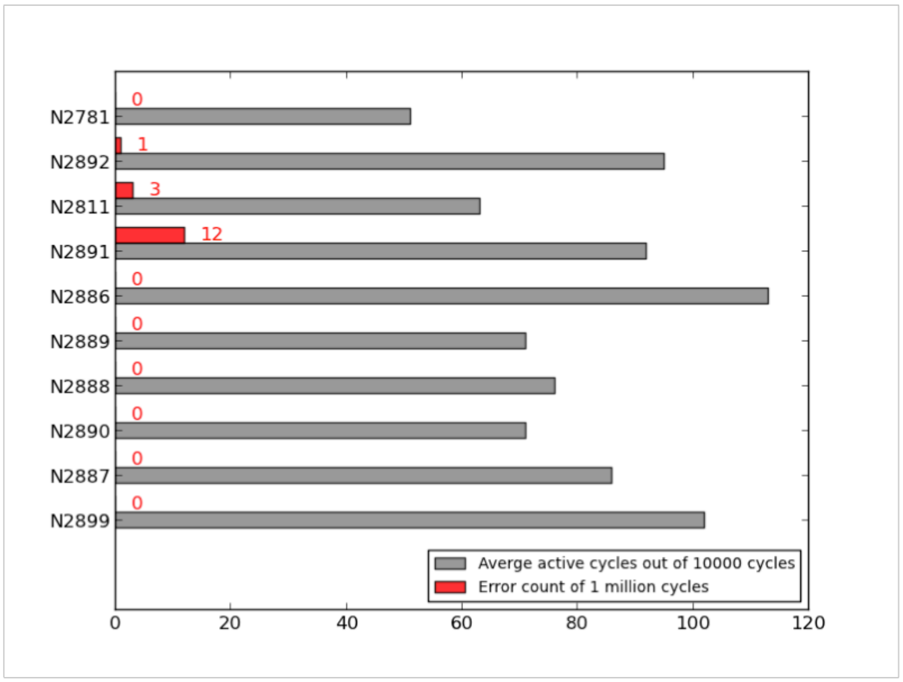


Figure 18: Benchmark C1908 outputs with the average active cycles out of 10,000 cycles (using 100 simulation trials), and the total error counts for 1 million cycles at clock period of 1.7 ns.

Determining the real error-contributing output based only upon the output's activity rate is misleading, because errors only occur when the PO settles after the required clock period. We need to have the settling behavior of error-possible outputs to estimate the error rate for the given workload, and then a prediction can be made for the error-contributors.

Therefore, an analysis of the settling time of each PO is necessary to identify the error-contributor POs, thus Category IV is added:

IV. **Error-contributor-POs:** The worst case to the output that has a longer propagation delay than the clock period, and is highly likely to settle after the required clock period.

Setting the threshold value of stabilization probability to identify Category IV POs depends upon the level of error tolerance of the EDAC module that will be incorporated into the design. The stabilization probability is directly linked with the settling time for each cycle.

Error Count Estimation And Error Rate Calculation

Based on the settling time histogram, one can calculate the stabilization probability and predict the error rate for all POs and all possible clock frequencies. In this section, the detailed method of how to obtain the error count and to calculate the error rate is discussed.

Figure 19 shows the settling time histogram of all error-possible POs for Benchmark C1908. The settling time histogram of the desired outputs can be plotted with an appropriate bin size. In this work, the bin size is 50 ps, because it is the average propagation delay of a NAND gate for the Synopsys 32-nm library. The histogram is plotted based on the .vcd file of original clock period. The error estimation rate is calculated for each small range of clock frequency based on the bin size. In this work, the error estimation essentially matches the simulation result because 50 ps is also the step size of the swept simulation as in Chapter V.

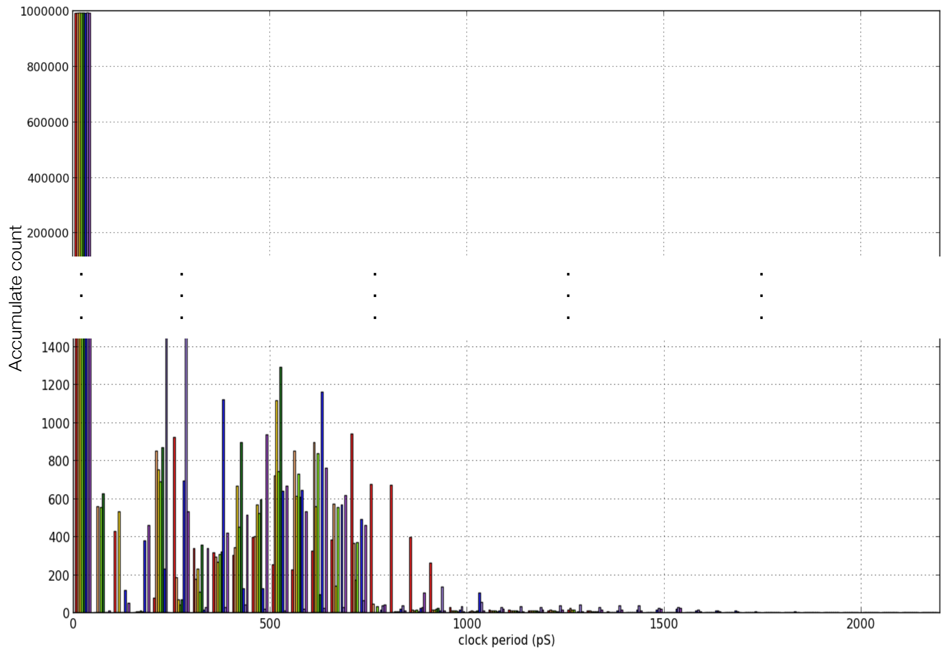


Figure 19: Benchmark C1908 settling time histogram of each error-possible PO. The x-axis is the settling time in picoseconds, and the y-axis is the accumulated number within each bin.

For Benchmark C1908, most of cycles the POs settled at 0 ns, which means that most POs in C1908 are not active with the given input workload during the simulation time. Errors only occur during those cycles that settled after the required clock period. Figure 20 shows the stabilization probability curve of each PO; the thick black line is for the whole circuit. The stabilization probability curve is the cumulative distribution function (CDF) of the output's settling histogram. The error-contributing POs can be identified based on the stabilization probability curves.

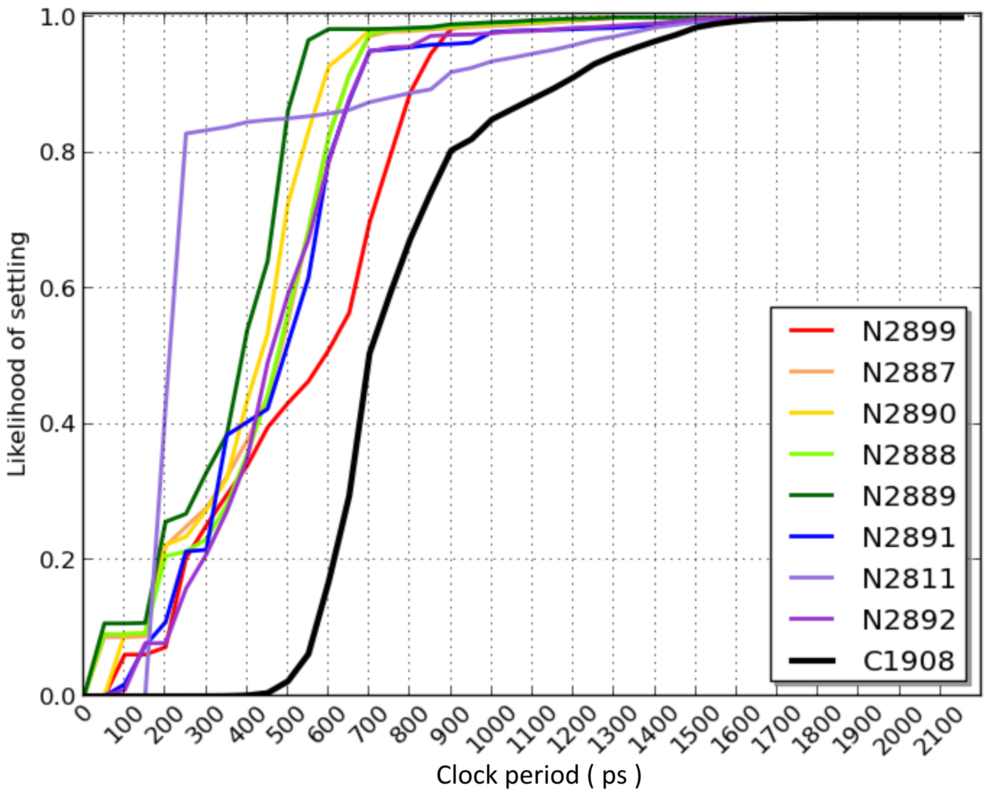


Figure 20: Benchmark C1908 stablization probability of each error-possble PO.

Continuing with the example of Benchmark C1908, if the operating clock period is 1.7 ns, for example, then outputs N2891, N2811, and N2892 are the error-contributor POs, because their stabilization probability did not reach 1 by 1.7 ns. Figure 21 shows the stabilization probability of error-contributor POs: N2891, N2811 and N2892 in full and zoom-in versions. The exact error count can be calculated by summing the histogram bins that stabilize later than the selected operating clock frequency. According to the histogram, the estimated error counts for output N2891 = 13, N2811= 4, N2892 = 1, while the simulation results of error count of output N2891 = 12, N2811 = 3, and N2892 = 1.

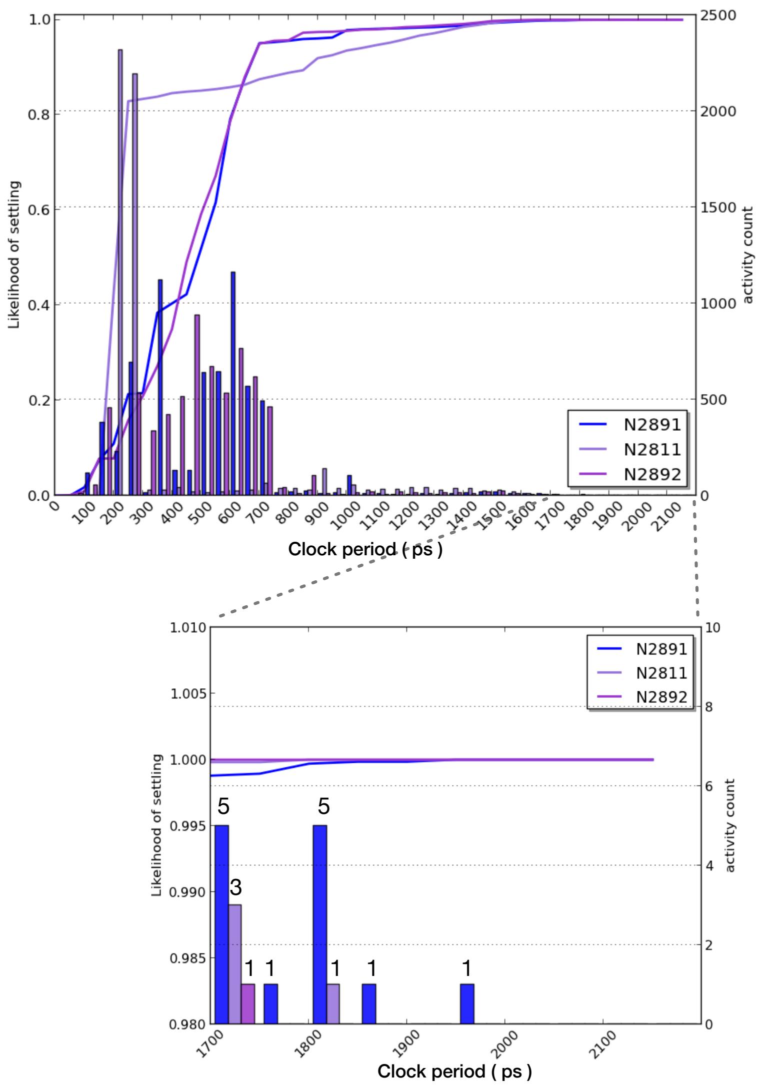


Figure 21: Benchmark C1908 settling time histogram and stabilization probability density function of outputs N2891, N2911, N2892.

Each PO has a different error rate with the same operating clock frequency. The formula used to calculate the error rate:

$$Error\_rate = 1 - Stabilization\_probability \qquad (1)$$

Error Estimation Results Discussion

As discussed previously, the static longest path does not always impact the observed errors of the circuit. It may not be active for a certain input workload. When evaluating the capabilities of existing commercial EDA tools, it was found that they did not directly provide path activity information. As part of this dissertation work, a method was developed to obtain each output's settling behavior curve by analyzing the .vcd file. The error count and the error rate trend of each output is predicted.

A circuit could have multiple outputs, but only the Category II - error-possible outputs - have the probability to experience errors. Therefore, analysis has performed on the output settling behavior of error-possible outputs in order to obtain the error rate estimation; this analysis enables the identification of the Category IV - error-contributing POs. Customized scripts extract the settling timestamp for every cycle of the tested outputs, and the histogram of the results indicate the error count of the tested outputs. The accuracy of this method is confirmed later with the simulation results. Each output has one histogram of the settling time probability.

Table 4 shows the longest delay for each of the Category II error-possible outputs for C432, C880, C1908 and C6288. The static critical delays of each benchmark are 2.41 ns, 2.01 ns, 2.20 ns, and 4.82 ns respectively. As an example, assume that the goal is to operate the circuit at 70% of original clock period, which corresponds to clock periods of 1.70 ns, 1.40 ns, 1.50 ns, and 3.40 ns respectively. Any output propagation delay that is longer than the operating clock period has the potential to generate errors. The maximum clock speculation explored in this work is 30% higher than the original, error-free design.

Notes that the desired performance improvement would ultimately be determined by the design team.

Table 4: Benchmark circuit static propagation delay of all error-possible outputs

C432	**N421**	**N432**	**N431**	**N430**	**N370**					
	2.41 ns	2.25 ns	2.24 ns	2.11 ns	1.75 ns					
C880	**N878**	**N866**	**N879**	**N880**	**N874**	**N863**				
	2.01 ns	1.89 ns	1.85 ns	1.82 ns	1.7 ns	1.48 ns				
C1908	**N2899**	**N2887**	**N2890**	**N2888**	**N2889**	**N2886**	**N2891**	**N2811**	**N2892**	**N2781**
	2.20 ns	2.18 ns	2.18 ns	2.18 ns	2.17 ns	2.15 ns	2.07 ns	1.97 ns	1.81 ns	1.67 ns
C6288	**N6288**	**N6287**	**N6280**	**N6270**	**N6260**	**N6250**	**N6240**	**N6230**	**N6220**	**N6210**
	4.82 ns	4.74 ns	4.7 ns	4.57 ns	4.44 ns	4.32 ns	4.19 ns	4.06 ns	3.93 ns	3.81 ns

To obtain the actual error count, the settling histogram of each output gives more direct and detailed information. Figure 22 shows the settling time histogram of each tested benchmark circuit with the original clock period. Because of the large number of inactive cycles for some benchmarks, the histogram in Figure 22 did not include the inactive cycles, and the y-axis is different in scale for different benchmarks to display the settling behavior better. The error rate calculation is based on the stabilization probability curve, which contains the inactive cycles. The two vertical lines on each graph in Figure 22 mark the 70% of original clock period and the latest settling time on record. The latest settling time for circuit C421 is 2.35 ns, for C880 is 1.90 ns, for C1908 is 1.95 ns, and for C6288 is 4.00 ns.

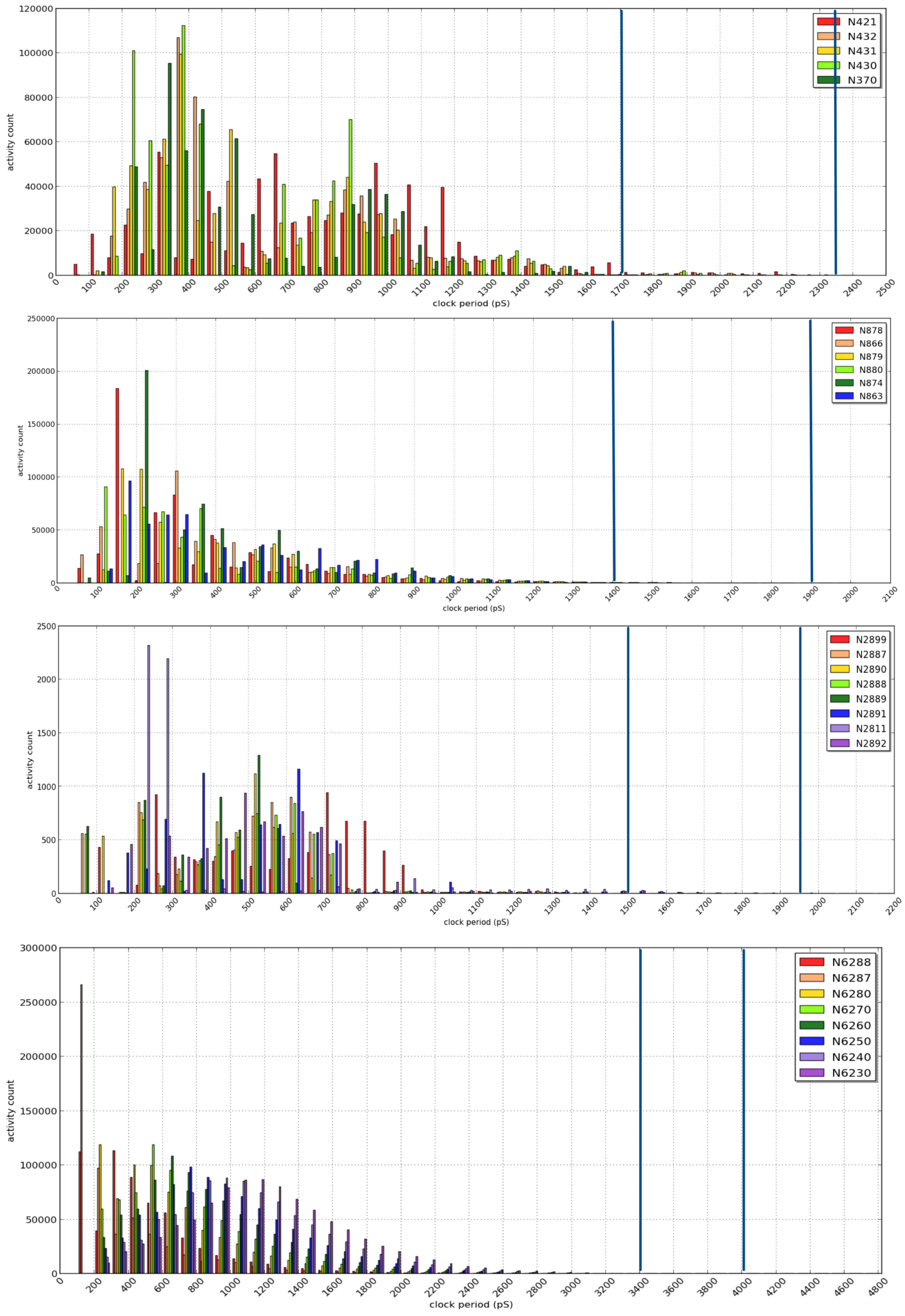


Figure 22: Outputs settling time histogram with 1 million random input vectors of four benchmark circuits C432, C880, C1908 and C6288. The x-axis is the clock period. The y-axis is the activity count. The right vertical line represents the maximum recorded settling time. The left vertical line represents 70% of the original clock period.

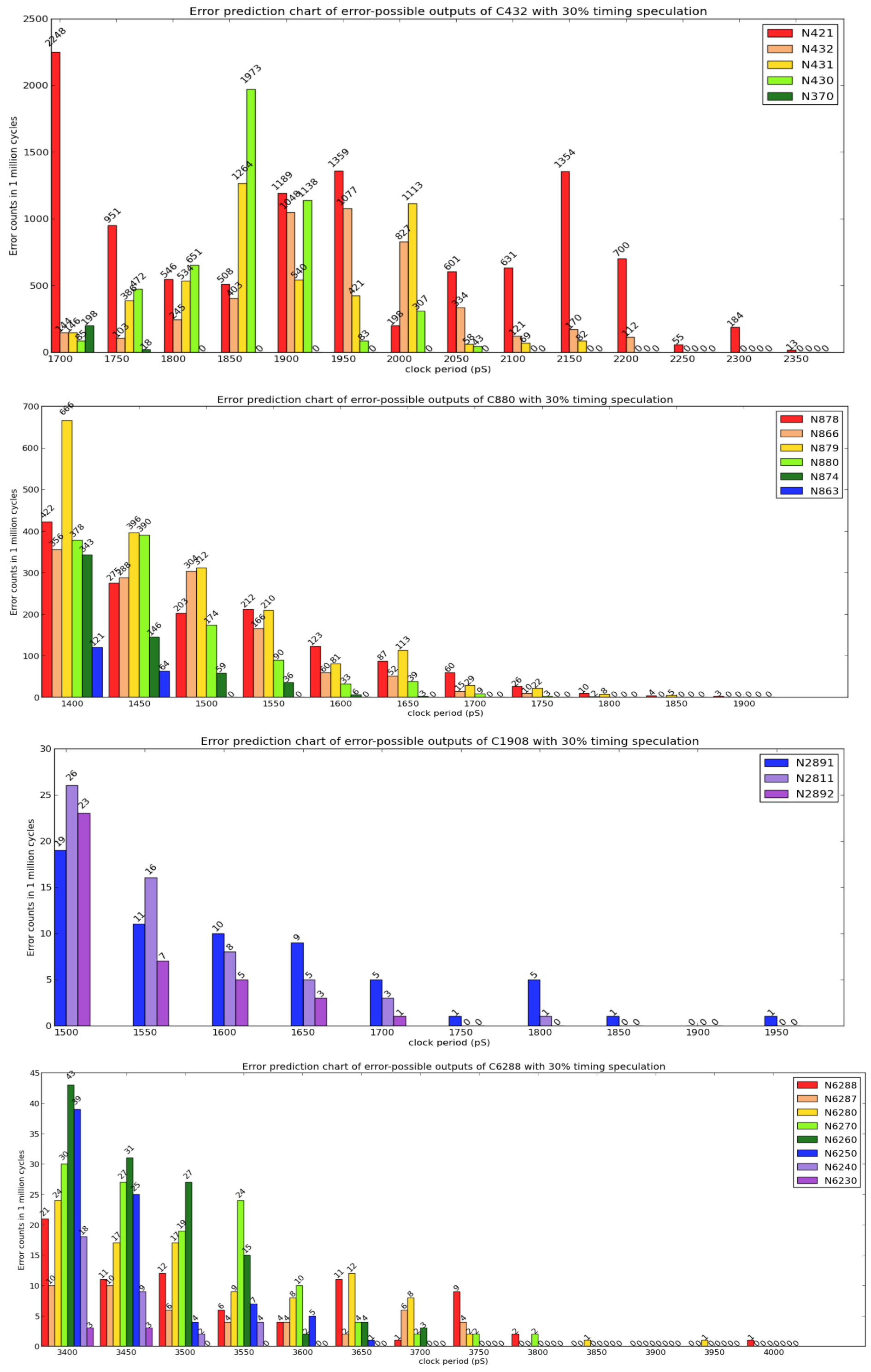


Figure 23: Zoomed settling time histogram between 70% of orignal clock to the error-free clock of four benchmark circuits. The x-axis is the clock period. The y-axis is the error count observed in 1 million cycles.

Figure 23 shows the zoomed view of the area between those two marked lines. Each of the error possible outputs is shown in Figure 22 and Figure 23, and the legends are in descending order of the propagation delay from top to bottom. According to the data in Figure 23, the output that contained the longest path will be the first output to observe errors when the operating clock period is reduced. However, if the operating clock period is reduced further, then the dominant error contributor may change to the PO that settled more frequently after the selected clock period.

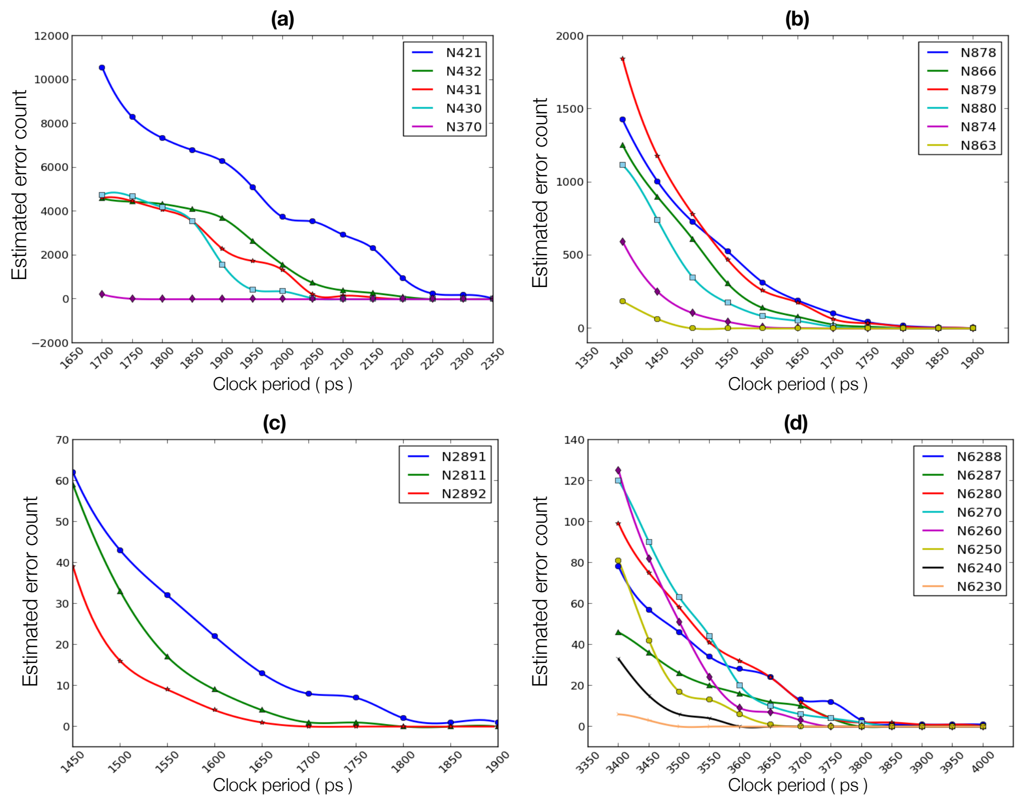


Figure 24: Estimated error count of each error possible outputs of (a) C432, (b) C880, (c) C1908, and (d) C6288, from 70% of original clock period to the error-free clock period.

Figure 24 displays the error count estimation out of 1 million cycles of each error-possible PO from 70% original clock period to the error-free clock period. The largest error-contributing PO changes with the operating clock period. Errors may concentrate to certain POs, or they may evenly distribute across different POs. For Benchmark C432, the static longest output, N421, is the most error-prone output for all clock periods. For Benchmark C880, the most error-prone output shifts from the static longest path, N878, to N879 when the clock period is reduced to 75% of the original clock period. The same observation is made for Benchmark C6288, the largest error-contributing PO shifts to a lower bit along with a reduction in clock period. This result is because of the ripple structure of the multiplier, where results in the higher-order bits depend on the lower-order bits.

Table 5: Comparison of error composition from critical path and greatest error-contributing PO other than the critical path.

	C432	**C880**	**C1908**	**C6288**
Operating clock period	1.7 ns	1.4 ns	1.5 ns	3.4 ns
Errors from critical path	29.7%	21.9%	0%	13.7%
Errors from the greatest error contributor other than critical path	23.2%	28.5%	37.3%	22.1%
Total errors	20,455	6,400	158	562

Table 5 takes 70% of the original clock period as an example to show the errors from the static critical path and the total errors. For the tested four benchmark circuits, only C432's critical path is the largest error-contributor PO. The application of an error-reduction method on the identified error-contributor POs specifically will help to reduce errors effectively. In this work, it is assumed that the error-tolerant threshold value is

0.1% (i.e., total error count < 1,000) for the whole system. The targeted timing speculation level is 70% of the original clock period. The error reduction method and results will be discussed in detail in the next chapter.

Conclusion

In summary, there are three steps of the all-clock-frequency error estimation method in this work:

1. Identify POs in Category II based on the static timing delay.
2. Extract each PO's settling time for each cycle from the .vcd file.
3. Plot the settling time histogram of each PO and calculate the probability density function to estimate each PO's stabilization probability.

In this work, a PO is considered as an error-contributor (Category IV) if the error rate of the PO is twice the average error rate of the whole circuit. Therefore, if the operating clock period is set at a speculation level of 70% of the original clock period, then the steps to identify Category IV POs are: (1) Find out all error-possible POs according to the static propagation delay; (2) Calculate (estimate) each error-possible PO stabilization probability at the targeted speculating clock period; then (3) Determine that according to the error rate calculation formula, the $Stabilization\_probability = 1 - error\_rate$. This stabilization probability value is the threshold value for identifying Category IV error-contributor POs. Figure 25 shows the stabilization probability curve of the four benchmark circuits for the given input workload. Designers can then identify the error-contributing POs for a desired clock period.

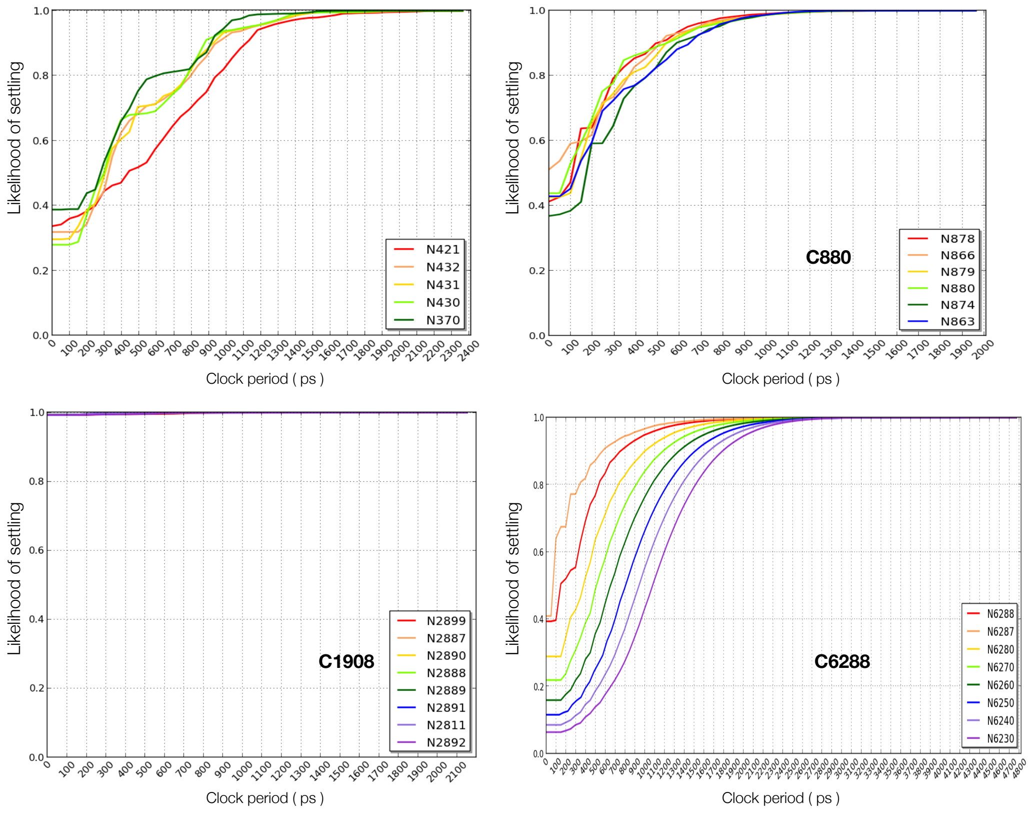


Figure 25: The stabilization probability of four tested benchmark circuits for the given workload from start to end of the clock period. The x-axis is the clock period. The y-axis is the likelihood of settling.

CHAPTER V

OFF-LINE ERROR CHECKING METHOD

As discussed in Chapter I, many kinds of error detection circuits exist, which require special circuitry for parallel comparison. The proposed error-checking module was developed to perform statistical analysis of errors after simulation in order to inform the mitigation approach and to confirm the error estimation results.

Approaches For General Error-Checking And Off-Line Error-Checking

The regular error checking method requires either the golden circuit or the delay element in the simulation to detect errors and to preserve correct results and detect errors. Therefore a special wrap circuit needs to be modified accordingly for every DUT. Figure 26 and Figure 27 show the general structure of two existing error-checking methods, and Figure 28 shows the off-line error-checking method used in this work.

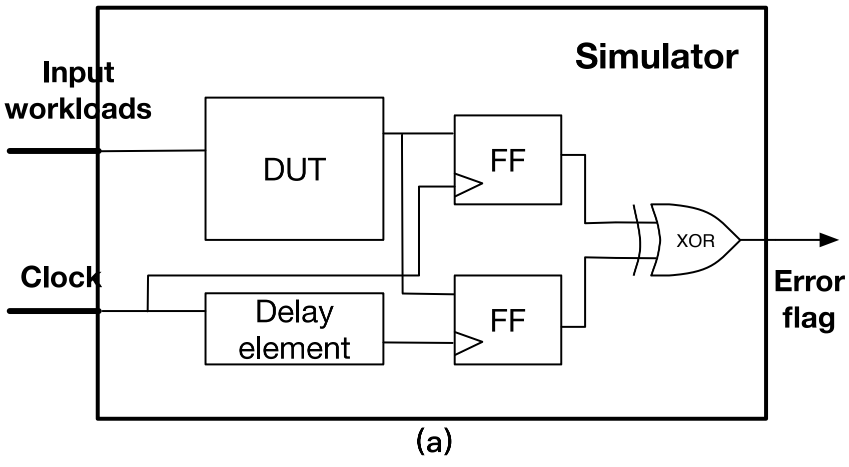


Figure 26: The general structure of (a) transition detection method

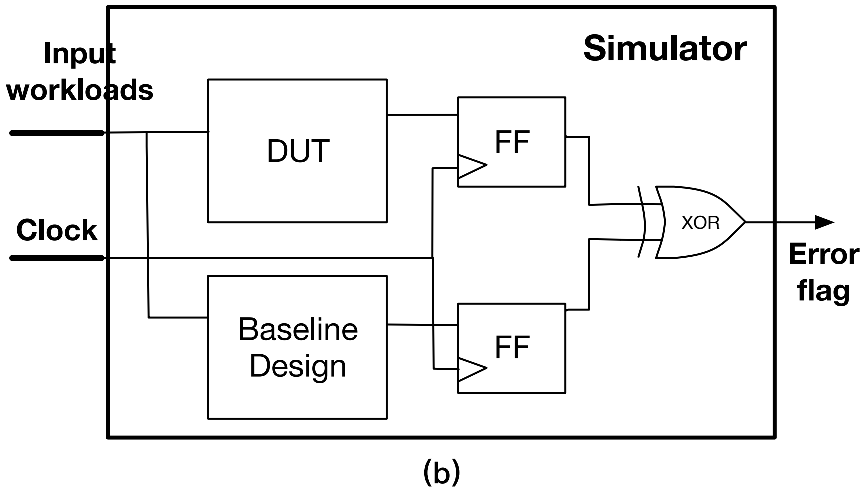


Figure 27: The general structure of (b) duplication module/path method

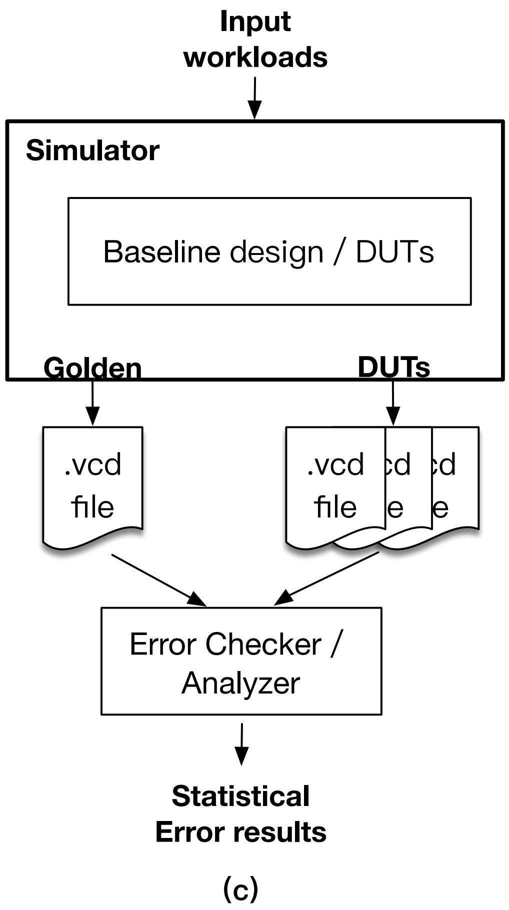


Figure 28: The general structure of (c) proposed off-line error checking method.

The off-line error checking method uses customized Python scripts to extract information from the value change dump (.vcd) file, and compares the settled value of each output between golden copy and the tested one after the simulation. To detect errors, each output's settled value at the end of each cycle is extracted from the tested circuit's .vcd file; a comparison is made with the golden results cycle by cycle. Both the saved golden .vcd file and the tested ones can be used multiple times for different analysis and comparison. This off-line error checking method enables the possibility of on-demand post-simulation error analysis and saves on run time. This proposed error detection method does not need a customized test-bench, and cell activity analyses are all based on extracting and processing the .vcd file.

Reformatting The .vcd File For Error-Checking

The .vcd file saves complete information during the simulation in the body part. However, only the PO's final status of each cycle is important to implement the offline error-checker. Reformatting the raw .vcd file information is the first step to implement the off-line error-checker.

Figure 29 is the algorithm flow chart of the data preparation scripts. After reading in the original text of the .vcd file, a customized Python script is used to restructure it into the appropriate formation to later process. The header part will be removed, and the switching nodes status of each cycle is saved in one entry without the timestamp. Figure 30 shows an example of the extracted activity file. Each entry records the status for all nodes for one clock cycle without timestamps. The Appendix Part B (1) shows the example code to prepare the activity information from raw .vcd files to a formatted txt file for further usage. The bash scripts to automate the process were not included in the Appendix.

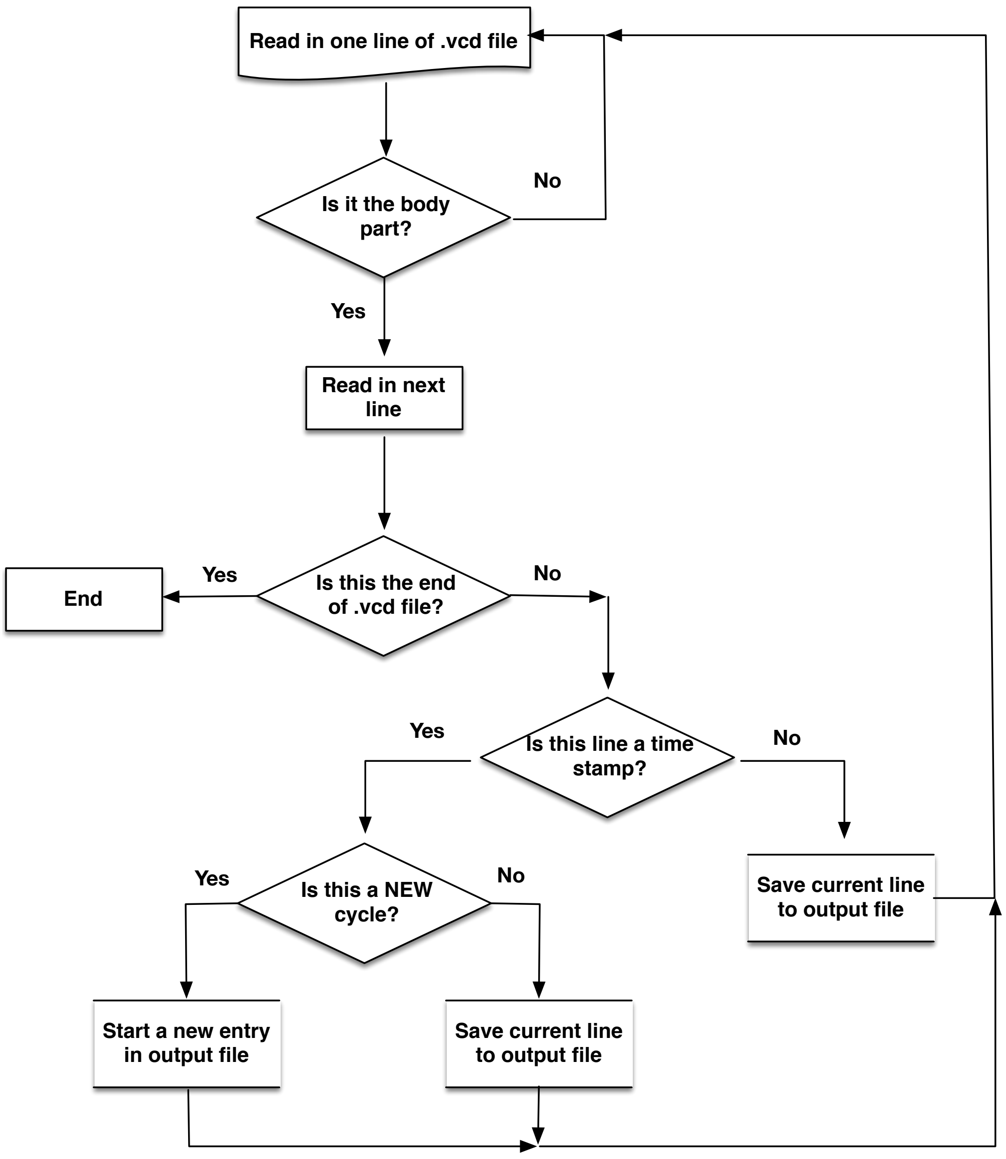
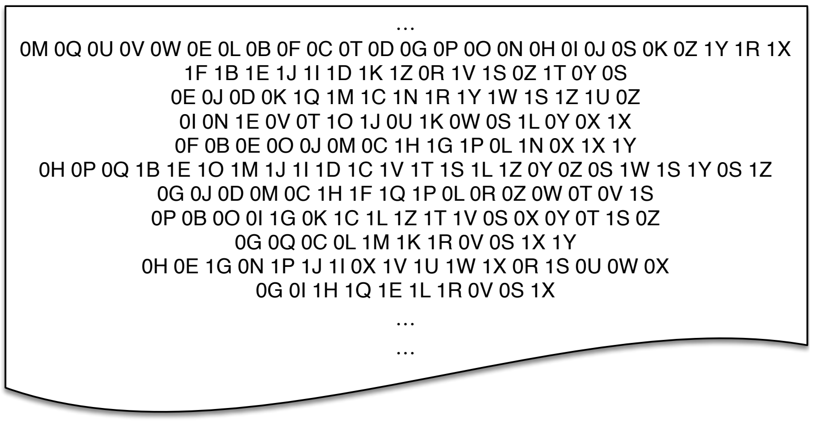

Read in one line of .vcd file
Is it the body part?
No
Yes
Read in next line
Is this the end of .vcd file?
Yes
End
No
Is this line a time stamp?
Yes
No
Is this a NEW cycle?
Yes
No
Save current line to output file
Start a new entry in output file
Save current line to output file

Figure 29: Algorithm flow chart of data preparation script.



```
…
0M 0Q 0U 0V 0W 0E 0L 0B 0F 0C 0T 0D 0G 0P 0O 0N 0H 0I 0J 0S 0K 0Z 1Y 1R 1X
1F 1B 1E 1J 1I 1D 1K 1Z 0R 1V 1S 0Z 1T 0Y 0S
0E 0J 0D 0K 1Q 1M 1C 1N 1R 1Y 1W 1S 1Z 1U 0Z
0I 0N 1E 0V 0T 1O 1J 0U 1K 0W 0S 1L 0Y 0X 1X
0F 0B 0E 0O 0J 0M 0C 1H 1G 1P 0L 1N 0X 1X 1Y
0H 0P 0Q 1B 1E 1O 1M 1J 1I 1D 1C 1V 1T 1S 1L 1Z 0Y 0Z 0S 1W 1S 1Y 0S 1Z
0G 0J 0D 0M 0C 1H 1F 1Q 1P 0L 0R 0Z 0W 0T 0V 1S
0P 0B 0O 0I 1G 0K 1C 1L 1Z 1T 1V 0S 0X 0Y 0T 1S 0Z
0G 0Q 0C 0L 1M 1K 1R 0V 0S 1X 1Y
0H 0E 1G 0N 1P 1J 1I 0X 1V 1U 1W 1X 0R 1S 0U 0W 0X
0G 0I 1H 1Q 1E 1L 1R 0V 0S 1X
…
…
```

Figure 30: The example of partial activity file extracted .from .vcd file.

Implementation Of The Off-Line Error-Checker

After the raw .vcd file is processed, the formatted activity text file of both the test circuit and the golden copy are read into the Extract and Compare script. A LUT (Look Up Table) is pre-defined with the selected nodes of interest and the notation used in the .vcd file. For each cycle, the switching activity of the selected nodes are saved into the LUT in sequence. Although one node could change multiple times within one cycle, only the final (i.e., settled) value is directly related to the correctness of operation. Therefore, a subsequent switching activity in the cycle always overwrites the previous one in the LUT. The scripts first identify whether any switching activity of selected nodes are contained in this cycle, and then the LUT is modified accordingly. Then, a comparison is made between the test LUT and the golden LUT; the error count is increased accordingly for each PO whenever it detects a cycle that does not match. Figure 31 shows the algorithm flow chart for the Extract and Compare script that is given in Appendix Part B (1). The bash scripts to automate the process were not included in the Appendix.

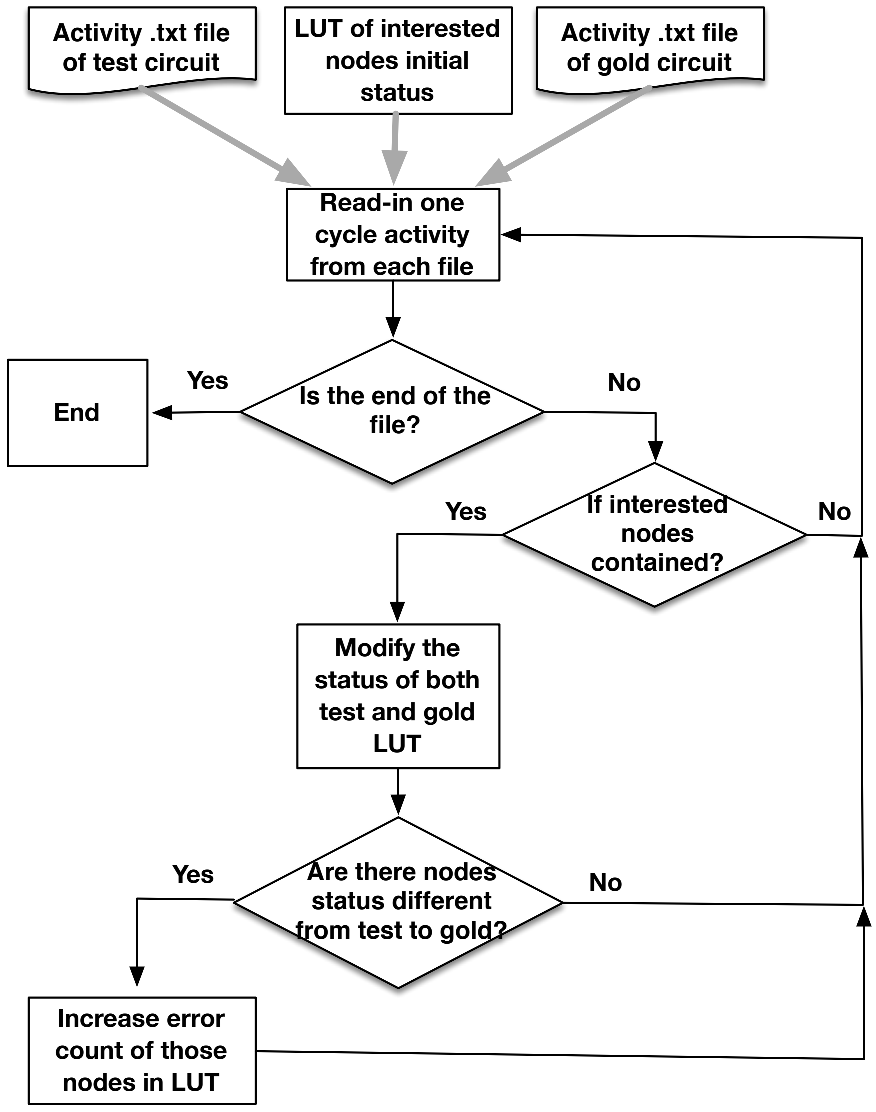

Activity .txt file of test circuit
LUT of interested nodes initial status
Activity .txt file of gold circuit
Read-in one cycle activity from each file
Is the end of the file?
Yes
End
No
If interested nodes contained?
Yes
No
Modify the status of both test and gold LUT
Are there nodes status different from test to gold?
Yes
No
Increase error count of those nodes in LUT

Figure 31: The flow chart of the Extract and Compare script's algorithm.

Comparison Of Error Estimation vs. Error Checking Results

According to the error estimation method introduced in Chapter V, there are error count estimation results for all clock frequencies. This section presents the results of the simulation data of the error count. The accuracy of the error estimation method is confirmed by simulation data.

For the evaluation, the circuits are synthesized, placed, routed, and simulated with Synopsys tools and the Synopsys 32-nm library. The simulation clock period is swept from the error-free clock period to 70% of the original clock (i.e., the static critical path delay time). The simulation step size is 50 ps, however, Figure 32 show step size is 0.1 ns for a clear view in graph.

Figure 32 shows the error trend with the decrease of the clock period and the comparison of simulation and prediction errors counts. With the knowledge of each output's settling information, designers could select a speculative clock period at an acceptable error rate tolerance. Because the error estimation method is made based on settling time histogram of each output with original clock period.

For the binning procedure, the larger the bin size, the less the total bin number. If the bin size increased from 50 ps to 100 ps, the bin number will be halved. Since the sampling data stay the same, therefore the value of each bin will change. However, whether the bin size will affect on the estimation results is depending on the targeting clock period precision and the bin size. As long as the bin size is smaller than the targeting clock period precision, the estimation results will not be affected, because the estimation is calculated by the sum of bin value that settling time above the targeting clock.

For example, if we are trying to estimate error count for clock period 1.8 ns (precision at 100 ps). Changing bin size from 50 ps to 100 ps will not affect the estimation results, since all the sampling data over 1.80 ns will fall into the bins that counted as error. On the other side, if we are trying to estimate error count of 1.85 ns (precision at 50 ps) clock period, changing bin size from 50 ps to 100 ps, it will lead to some ambiguous sampling point of bin (1.80 – 1.90) for error estimation. For instance, if there is a clock cycle, an output settled at 1.82 ns (no error) and 1.88 ns (error) will fall into one bin.

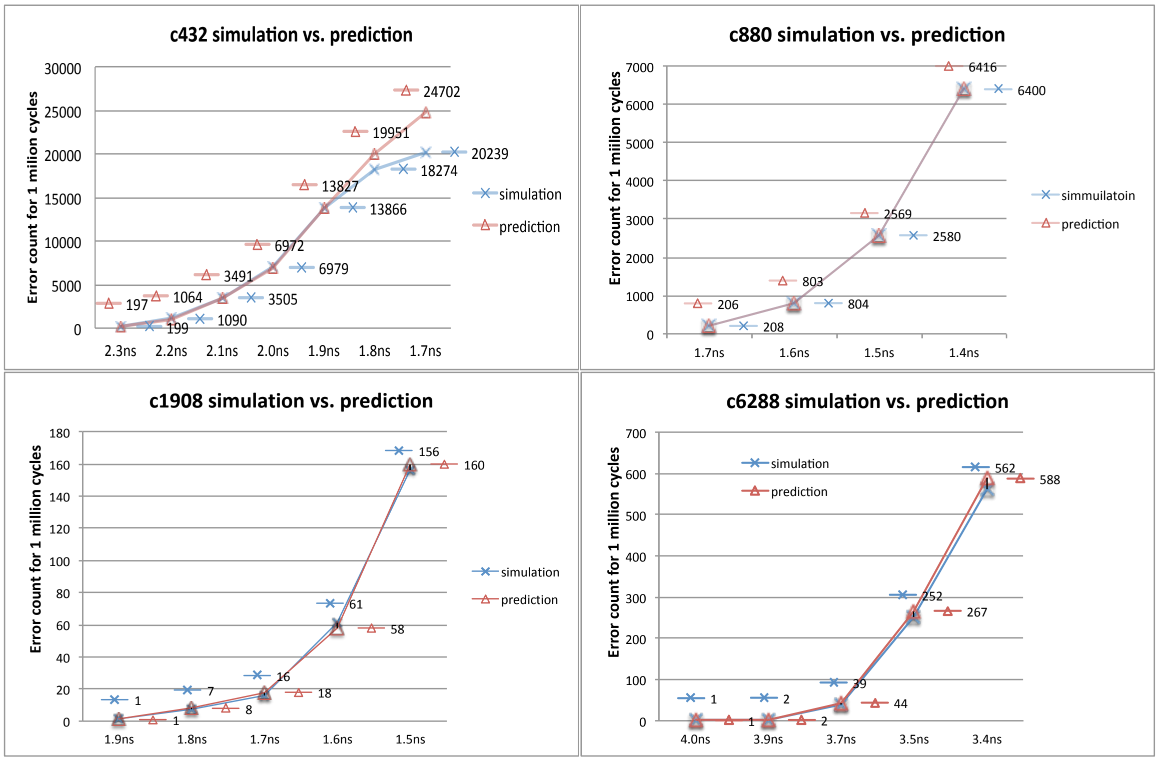


Figure 32: The comparison between simulated results and total error estimation trends of four tested benchmark circuits.

Conclusion

The off-line error-checking method allows a designer to check errors and perform specific analysis after simulation, which suits the demand in this work perfectly. The Python implementation of the error-checking module was developed to process the raw .vcd file, so that it can be used universally on all types of circuits, and it does not require any test wrap circuit during simulation. To ensure the correctness of error estimation, the bin size should not be larger than the clock period precision during circuit behavior curve statistical analysis.

CHAPTER VI

DUAL-THRESHOLD VOLTAGE APPROACH FOR TIMING ERROR REDUCTION

In this work, a dual-threshold voltage approach is used on selected cells to improve the propagation delay of identified error contributing POs. For the given input workload in this work, Category IV - the real error-contributing POs – have been identified in Chapter IV. However, the fan-in cone contains multiple paths that feed into a PO. Replacing all the cells on the fan-in cone is impractical due to the leakage power increase of using Low-Vt cells. Therefore, consideration must be given to improving error rate in a cost-effective manner.

Dual-Threshold Voltage Approach For Re-Timing

The Synopsys SAED_EDK 32/28_CORE digital standard cell library [57] was used in this work. The library includes typical miscellaneous combinational and sequential logic cells for different drive strengths. It also contains cells with different versions (multi-voltage, multi-threshold, etc.) for low power designs. In order to implement multi-threshold low power techniques High-V_t (HVT), Low-V_t (LVT) and Standard-V_t (SVT) versions of the library was created.

Multi-threshold / Dual-threshold technology mostly uses to reduce leakage power by using the HVT cells whenever performance goals allow, and in this work, the LVT cells will be used to where necessary to meet timing.

According to the analysis of circuit typical case timing behavior, selected cells to be replaced with LVT version in the net-list file generated synthesis. Because the modification of the net-list did not structurally change the circuit connection, there are minimum impacts on the circuit activity behavior. The error-contributing POs timing closure improved as desired to specifically reduce timing errors. The steps to implement the dual-threshold voltage approach to improve certain paths/ POs delay are listed:

1. List all error-possible POs.
2. Identify the error-contributing POs using the error-estimation method.
3. List the fan-in paths that have longer propagation delay than the clock period.
4. Identify the convergence point of paths listed in Step 3.
5. Replace cells after the convergence point with low-V_t cells.
6. Perform cell activity analysis as described previously on the remaining cells.
7. Weight each cell's activity level.
8. If the activity level is higher than 50%, then replace the cells with low-V_t cells.

The detail of critical cell identification is introduced in next section.

Identification of Critical Cells

The goal of this work is to reduce the error rate more efficiently by shortening the propagation delay of the identified error-contributing POs. With the knowledge of the circuit behavior under a given input workload, there are two competing objectives that need to be met: reducing more errors while using fewer Low-V_t cells.

Finding the right cells to replace is the key to reduce errors effectively in this approach. After identified error-possible PO, we have to working on the whole fan-in cone of the PO to select cells to replace.

The critical cells as defined in this work can be categorized into three types:

A. Stem cells (Green): the cells after the convergence point with an active rate greater than 50% for the given workload.

B. Shared cells (Yellow): the cells shared by more than one branch or shared by other fan-in cones with an active rate greater than 50% for the given workload.

C. Highly active branch cells (Blue): the cells only used by one fan-in cone, but have an active rate greater than 50% for the given workload.

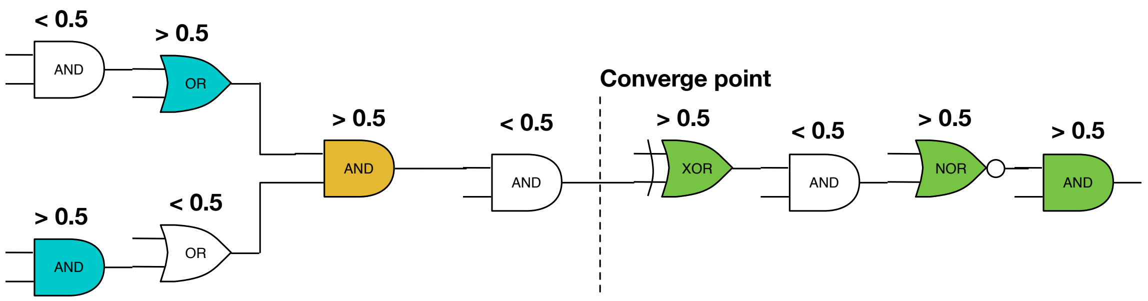


Figure 33: A partial circuitery to differentiate three types of critical cells that are going to be replaced in this work.

Figure 33 shows an example of the three types of critical cells in a fan-in cone to give a more intuitive definition. For C1908, the traditional critical path leads to the PO N2899 with 2.20 ns propagation delay. The identified error contributor POs are N2891, N2892, and N2811. Take PO N2891 as an example, the longest 5 paths to the PO are from inputs: N4, N1, N7, N13, N19. These 5 inputs are the start point of longest paths of PO N2899, N2887, N2890, N2888, N2889, N2886, N2811, N2892. The cell selection After identifying the stem cells and the shared cells, the activity of each cell was analyzed to determine replacement selection.

Figure 34 shows the activity level of cells on the traditional critical path. The cells activity analysis is similar as the output activity process. First, the activity of those selected cells from the golden .vcd file are extracted using the same algorithm that was used to extract the activity of POs for circuit stabilization probability analysis in Chapter IV. However, the representation symbol of selected cells is needed to update in side of testing script.

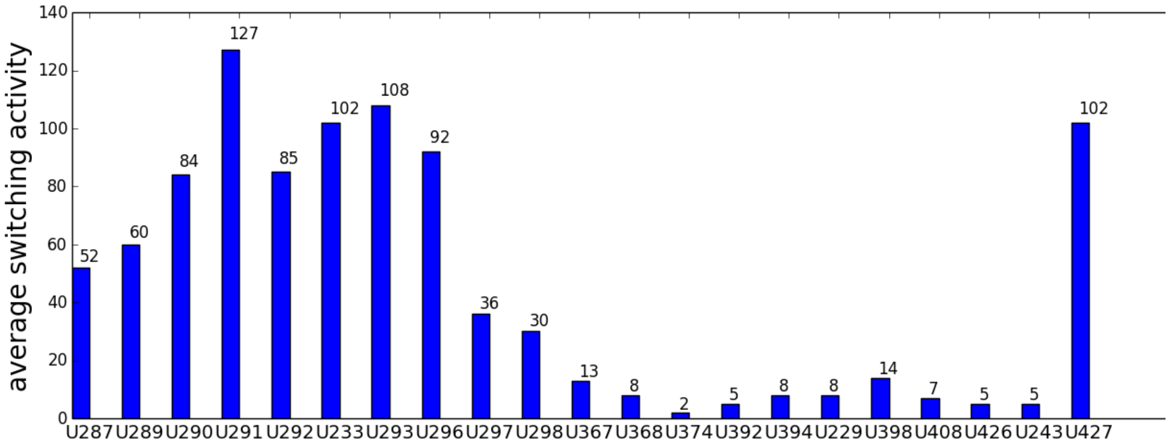


Figure 34: Benchmark C1908 critical path's cells activity.

Error Reduction Results Comparison And Discussion

To evaluate the effectiveness of the error reduction results, each circuit has three versions of implementation:

1. **Baseline Circuit** – uses the standard threshold voltage cells for all cells.
2. **Full-Path Replacement (FPR)** – replaces all cells on the static longest path with low-threshold voltage cells.
3. **Selected Cell Replacement (SCR)** – replaces cells selectively on the fan-in cone of the identified error-contributing POs with low-threshold voltage cells based on activity level.

The Full-Path Replacement represents the method that did not include the knowledge of circuit behavior for the given input workload. The Selected Cell Replacement represents the method that has benefited from understanding the circuit's typical behavior.

In this section, the error rate and improved error rate were compared at 70% of the original clock period. The maximum error-free clock frequency speed up was also compared between two methods.

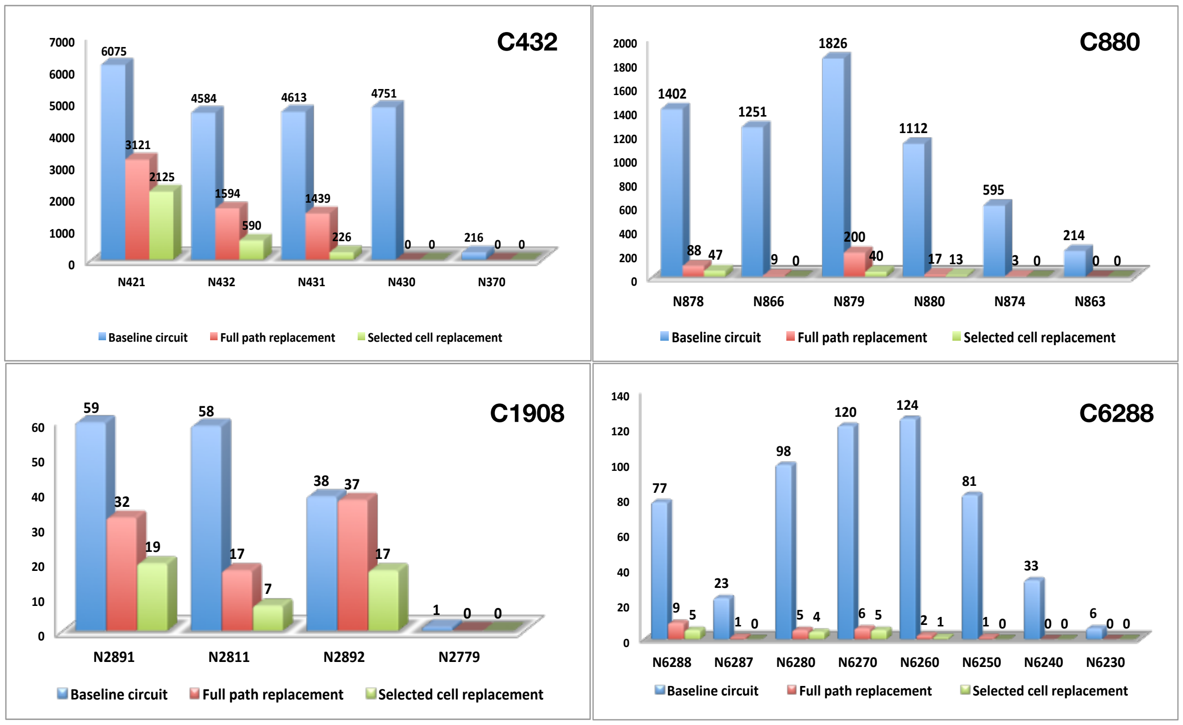


Figure 35: Error counts of each error-possible PO before and after error reduction method Full Path Replacement and Selected Cell Replacement. The operating clock period of C432 is 1.7 ns (70% of 2.41 ns), C880 is 1.4 ns (70% of 2.01 ns), C1908 is 1.5 ns (68% of 2.2 ns), and C6288 is 3.4 ns (70% of 4.82 ns).

Figure 35 shows four tested circuits with error comparison of three different the implementation. For each circuit, all error-possible POs are listed in the figures in descending order of propagation delay, from left to right. During the analysis, we observed that some POs would not generate any error even if their propagation delay was longer than the operating clock period. The error count numbers are marked on top. According to the results, the Selected Cell Replacement reduces more errors than the Full Path Replacement in general. The error reduction will be more obvious if the static critical path PO is not the identified error-contributing PO. Also, the PO that is more error concentrated will be more responsive on the Selected Cell Replacement method.

For C432, the static critical path PO is N421, and it is also the identified error-contributing PO. The Selected Cell Replacement method's advantage on output N421 is diminished, because most of replaced cells are the same as the Full Path Replacement method.

For C880, N879 is identified as the largest error-contributing PO, while N878 is the static critical path PO. By using the Selected Cell Replacement method, 80% more errors have been removed just for output N879.

For C1908, N2891 and N2811 are identified as error-contributing POs. Their propagation delays are the 7th and 8th longest path respectively. The static critical path leads to output N2899, but it does not generate any errors for the given input workload (one million random vectors) with the tested operating clock period (70% of the original). The Selected Cells Replacement method removes 58.8% more errors for output N2811 and 40.6% more for output N2891.

For C6288, the Selected Cells Replacement removes 50% and 16.7% more errors for the identified error-contributing POs N6260 and N6270. Also, 37.5% more errors have been removed in total. The results are relatively low compared to the other circuits because of the special structure of this multiplier. The paths to error-prone output N6260 and N6270 are just a subset of the critical path to output N6288.

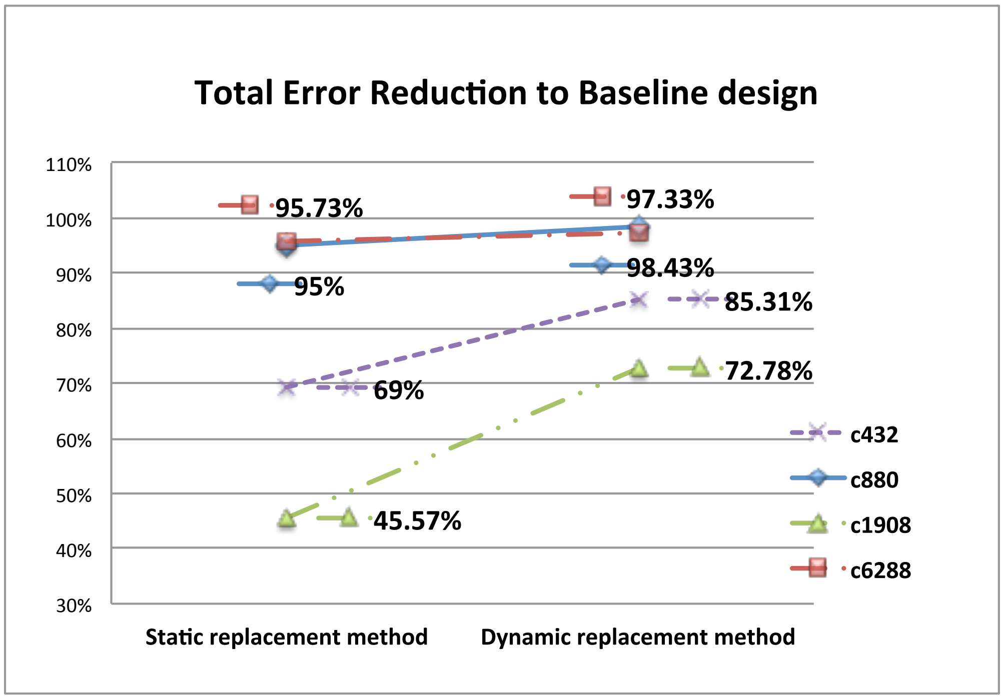


Figure 36: Total error reduction improvement from Full Path Replacement to Selected Cells Replacement, when operating at 70% of original clock period.

Table 6: Total error numbers comparison and the Selected Cells Replacement (SCR) improvement verses Full Path Replacement

	C432	C880	C1908	C6288
Full Path Replacement	6154	317	4	24
Selected Cells Replacement	2941	100	1	15
Improvement	52.2%	68.4%	75%	37.5%

Figure 36 shows error reduction results comparison between Full Path Replacement and Selected Cells Replacement. The actual total error counts and the improvement from the Full Path Replacement method to Selected Cells Replacement method are shown in Table 6. The Selected Cells Replacement method shows efficiency on error reduction when operating at 70% of the original clock period. Designers could select the timing speculation level based on the ability of the EDAC module. The error-free timing speculation clock is also tested. Figure 37 displays the speed increase comparison of Full Path Replacement and Selected Cells Replacement methods at the maximum error-free timing speculation clock period. Table 7 lists the Low-V_t cells usage in total number, and Table 8 shows the leakage power.

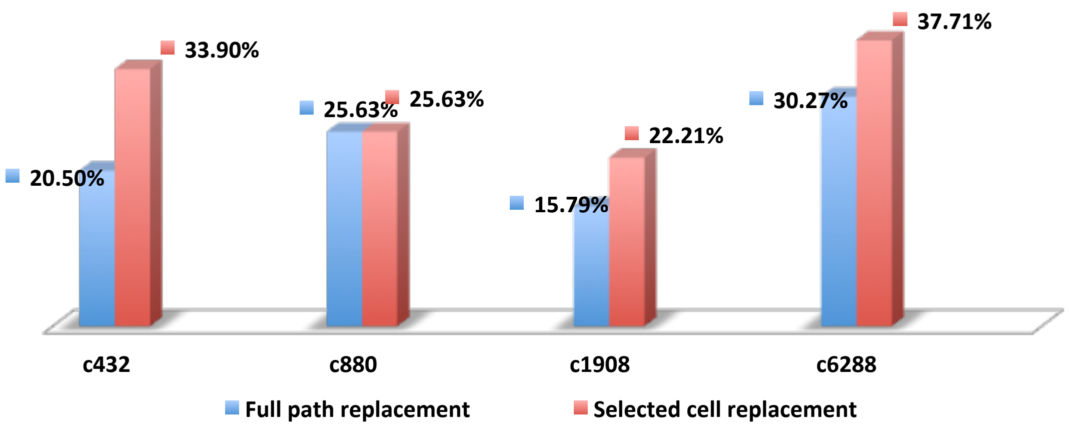


Figure 37: Error Free speed up comparison of Full Path Replacement method and Selected Cells Replacement method.

Table 7: Low-V_t cell usage comparison between Full Path Replacement (FPR) and Selected Cells Replacement (SCR)

	Total cell number	**Low-V_t cell number of FPR**	**Low-V_t cell number of SCR**
C432	81	18	6
C880	167	20	12
C1908	211	21	11
C6288	516	32	16

Table 8: Leakage power (µW) comparison of baseline, Full Path replacement (FPR) and Selected Cells Replacement (SCR)

	Leakage power of Baseline (µW)	**Leakage power of FPR (µW)**	**Leakage power of SCR (µW)**
C432	0.353	0.432	0.380
C880	0.953	1.04	1.00
C1908	1.33	1.40	1.39
C6288	0.546	0.832	0.717

<u>Conclusion</u>

This section compared the typical case workload behavior based timing optimization method Selected Cells Replacement (SCR) with baseline implementation and worst case (traditional critical path) timing optimization Full path Replacement (FPR) from many different aspects. Like the error reduction ability to a certain timing speculation percentage (30% up tested in this work), the tested all Benchmark circuits maximum error free operation point improvement of the SCR method, the Low-V_t cells ratios and leakage power dissipation.

CHAPTER VII

SUMMARY AND FUTURE WORKS

Higher chip performance is a constant demand in the semiconductor industry. The traditional design methodology sets a conservative guard band according to the worst case to ensure the correct operation. The impact of this constraint leads to lost performance. The BTWC design methodology optimizes a circuit based on the average case, and then it allows an error correction module to handle the errors. In this case, it is given the ability to cover the penalty for error correction by knowing the circuit's dynamic activity behavior for a given workload. Every error correction process has a penalty, therefore maximizing the operating clock frequency while controlling the total error counts leads to an overall gain. Based on the circuit's typical behavior under a given input workload, certain circuit timing paths be optimized to help effectively reduce errors during timing speculation.

This work introduced an error-estimation method for all-clock-periods without tedious simulations, and described a novel off-line error-checking method that does not require special test wrap circuit and simultaneous simulation. Both the error-estimation method and the error-checking method are based on extracted information from the raw .vcd file generated from simulating the circuit. The circuit's internal cell activity analysis is also obtained from the .vcd file. After understanding the circuit's dynamic activity under a given input workload, it can be re-synthesized with low-threshold voltage cells in the fan-in cones affecting the identified error-contributing outputs.

The results demonstrated that the error estimation method is accurate, and the error checking method provides a convenient way to detect and analyze errors for any PO. This error-reduction approach reduces a majority of errors while maintaining the minimum usage of low V_t cells. This work demonstrated the advantage of using the knowledge of the circuit's typical behavior and its impact on improving the performance and the error-reduction process.

The entire design flow was based on the typical commercial approach for synthesizing designs with standard cell libraries; the flow was augmented with customized Python scripts and is well incorporated with commercial EDA tools: Synopsys Design Complier, IC Compiler, Primetime and VCS.

The input workload used in this work was pure randomly generated input vectors on ISCAS85 Benchmarks. The random vector generator comes from Python library to modify the test bench stimuli. For a more realistic analysis, with the given benchmark circuit, apply typical application to the testbench and obtain circuit stabilization probability curve and error estimation to analyze the data with introduced methodology in this work.

This input workload variation caused timing behavior change has also been explored by Kevin E. Murray [58]. They introduced a new timing analysis formulation to form the circuit stabilization behavior with consideration of input combinations and compare the results with traditional SAT and Monte-Carlo simulation. Actually the big data analysis method could also be applied on circuit stabilization curve generation with regular usage of DUT. The stabilization curve will provide the insight of timing error estimation.

On the other hand, because of the reconfigurable character of field-programmable gate arrays (FPGAs), researchers could also explore the behavior analysis methodology described in this work on an FPGA board, and compare the performance improvements on timing, power, and errors.

APPENDIX

PART A:

(1) The script to extract the given PO all transition timestamps, each cycle has an entry.

```
#!/usr/bin/env python
# encoding: utf-8

import sys
import os
import string
import math
import numpy as np

ft = open(sys.argv[-2],'a') #### Define output file of interested POs
transition timestamps
f1 = open(sys.argv[-1],'r') #### Define input file of raw .VCD file
sbl=sys.argv[-3]            #### Define the interested PO's representing
symbol used in .VCD
p = sys.argv[-4]            #### Define the clock period
lines=f1.readlines()        #### Read in raw .VCD file
l = len(lines)              #### Get .VCD file length

i=0
for i in range (0,l):
    line = lines[i]
    if line[0] == '$':      #### Skip the header part
        continue
    if line[0] == '#':      #### if current line is a time stamp, Read-in
current line
        TimePoint = float(line[1:]
        residue_temp = TimePoint % int(p) #### Obtain the timing
status within this cycle
        if residue_temp != 0.0:        #### Check if it is a new cycle,
if not
            inner_count = i+1;
            line1 = lines[inner_count]     #### Continue reading the
next line of .VCD file
            while line1[0] != '#':         #### Check if this line is
a timestamp, if not,
                if line1[1] == sbl:        #### Check if this line is
the intrested node switching record, if yes,
                    ft.write(str(int(residue_temp))+" ")
#### Save current timestamp into the output file
                inner_count = inner_count+1
#### Read-in next line.
```

```
                if inner_count >= 1:
#### Chenk if the end of the .VCD file
                line1 = lines[inner_count]

        else:     #### if current timestamp is a new cycle, start a new entry in
the output file.
            ft.write("\n")
```

(2) The script to extract settling timestamp of every cycle for the given PO (with example of Benchamrk C1908, PO 2892).

```
#!/usr/bin/env python
# encoding: utf-8

import sys
import os
import string
import math
import numpy as np

ft
=open('c1908_output_N2892_rvt_2200_transition_time.txt','r')
fs
=open('c1908_output_N2892_rvt_2200_settling_time.txt','wb')

lines_ft=ft.readlines()
l_ft=len(lines_ft)
settle_time=[]

j=0
for j in range (0,l_ft):
    list_ft=lines_ft[j].split()
    if list_ft == []:
        continue
    else:
        settle_time.append(list_ft[-1:])
        fs.write(str(list_ft[-1:])+"\n")

print settle_time
fs.close()
ft.close()
```

PART B:

(1) The script to extract switched nodes for each cycle in sequence.

```
#!/usr/bin/env python
# encoding: utf-8

import sys
import os
import string
import math
import numpy as np

vcd_index = sys.argv[-1]
txt_index = sys.argv[-2]
f0 = open(sys.argv[-1],'r+')
tran = open(sys.argv[-2],'wb')
p = sys.argv[-3]
lines = f0.readlines()
l = len(lines)


i = 0
for i in range(0,l) :
    line = lines[i]
    if line[0] == '$':
        continue
    if line[0] == '#':
        TimePoint = float(line[1:])
        residue_temp = TimePoint % int(p)
        if residue_temp != 0:
            inner_count = i+1;
            line1 = lines[inner_count]
            while line1[0] != '#':
                tran.write(line1[0:-1]+" ")        #### Record the
switched node symbol
                inner_count=inner_count+1
                if inner_count>=l:
                    break
                line1 = lines[inner_count]

        else:
            tran.write("\n")
```

(2) The script to detect and calculate errors.

```
#!/usr/bin/env python

# encoding: utf-8


import sys

import os

import string

import math

import numpy as np

import matplotlib.mlab as mlab

import matplotlib.pyplot as plt


#Initialize primary outputs of golden copy and test copy; po_diff is the comparison
results for each cycle; er_output is the set showing errors for each cycle; er_count is
the accumilative error count of output.

po =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O
','P','Q','R','S','T','U','V','W','X','Y','Z','[','\\',']',
'^','_','`']      # Define symbol used in .VCD file to represent POs.

po_name =
['N545','N1581','N1901','N2223','N2548','N2877','N3211','N3
552','N3895','N4241','N4591','N4946','N5308','N5672','N5971
','N6123','N6150','N6160','N6170','N6180','N6190','N6200','
N6220','N6230','N6240','N6250','N6260','N6270','N6280','N62
87','N6288']     # Define the PO name of tested Benchmark
po_g =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0]     # Initial the golden copy’s LUT
po_t =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0]     # Initial the tested copy’s LUT
po_diff = []
gold_vcd = sys.argv[-1]
test_vcd = sys.argv[-2]
f0 = open(sys.argv[-1],'r+')
f1 = open(sys.argv[-2],'r+')
lines0 = f0.readlines()
lines1 = f1.readlines()

l = len(lines0)
i = 0
j = 0
k = 0
er_count =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0]
```

```
# compare settlized output of two copies for each cycle; text file is processed with
another python script, which sumarized internal transitions into one line one cycle.
for i in range (0,l):
    line_g = lines0[i]
    line_t = lines1[i]
    j = 0
    k = 0
    po_diff = [int(po_g[n])-int(po_t[n]) for n in
range(0,len(po_t))]  #find differences
    er_output=[abs(po_diff[m]) for m in
range(0,len(po_diff))]  #each output could have one error for each cycle
     er_count = [sum(x) for x in zip(er_count,er_output)]
     while j < len(line_g)-2:    # -2 because the last two symbol is '/n',
if do not remove, it will affect the iternation.
        if line_g[j] == ' ':
            j = j+1
        elif line_g[j] == '0':
            j = j+1
        elif line_g[j] == '1':
            j = j+1
        else:    #find output symbol in text file
            g_index = po.index(line_g[j])    #find the right index
in output list, and record current value in po_g
            po_g[g_index] = line_g[j-1]
            j = j+1


    while k < len(line_t)-2:
        #print j
        #print line_g[j]
        if line_t[k] == ' ':
            k = k+1
        elif line_t[k] == '0':
            k = k+1
        elif line_t[k] == '1':
            k = k+1
        else:
            t_index = po.index(line_t[k])
            po_t[t_index] = line_t[k-1]
            k = k+1
print er_count
#print zip(po,er_count)
#print zip(po_name,er_count)
er_rate = float(sum(er_count))/float(10000)
print er_rate


po_data = open('./po_data_mvt_2410.txt','a')
rate_data = open('./rate_data_mvt_2410.txt','a')
```

```
po_data.write(str(er_count)+'\n')

po_data.close()

rate_data.write(str(er_rate)+'\n')

rate_data.close()
```

REFERENCES

[1] S. Krishnamurthy, S. Paul, and S. Bhunia, “Adaptation to Temperature-Induced Delay Variations in Logic Circuits Using Low-Overhead Online Delay Calibration,” in *8th International Symposium on Quality Electronic Design (ISQED’07)*, 2007, pp. 755–760.

[2] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, “Scaling , Power , and the Future of CMOS,” in *IEEE International Electron Devices Meeting (IEDM 2005)*, 2005, pp. 9–15.

[3] S. Bhunia, S. Mukhopadhyay, and K. Roy, “Process variations and process-tolerant design,” in *20th International Conference on VLSI Design*, 2007, pp. 699–704.

[4] S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” in *IEEE Micro*, 2005, vol. 25, no. 6, pp. 10–16.

[5] S. Ghosh and K. Roy, “Parameter Variation Tolerance and Error Resiliency: New Design Paradigm for the Nanoscale Era,” in *Proceedings of the IEEE*, 2010, vol. 98, no. 10, pp. 1718–1751.

[6] P. Asenov, N. a. Kamsani, D. Reid, C. Millar, S. Roy, and A. Asenov, “Combining process and statistical variability in the evaluation of the effectiveness of corners in digital circuit parametric yield analysis,” in *The European Solid-State Device Research Conference (ESSDERC)*, 2010, pp. 130–133.

[7] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, “VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects,” *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 3–13, 2008.

[8] B. Greskamp, L. Wan, U. R. Karpuzcu, J. J. Cook, J. Torrellas, D. Chen, and C. Zilles, “Blueshift: Designing processors for timing speculation from the ground up.,” in *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA 2009)*, 2009, pp. 213–224.

[9] V. Mehrotra and D. S. Boning, “Modeling the effects of systematic process variation on circuit performance,” Massachusetts Institute of Technology, 2001.

[10] L. Xie and A. Davoodi, “Post-Silicon Failing-Path Isolation Incorporating the Effects of Process Variations,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 31, no. 7, pp. 1008–1018, Jul. 2012.

[11] K. Bowman, J. Tschanz, C. Wilkerson, S.-L. Lu, T. Karnik, V. De, and S. Borkar, “Circuit techniques for dynamic variation tolerance,” *46th Annu. Des. Autom. Conf.*, pp. 4–7, 2009.

[12] S. K. Nithin, G. Shanmugam, and S. Chandrasekar, “Dynamic voltage (IR) drop analysis and design closure: Issues and challenges,” in *11th International Symposium on Quality Electronic Design (ISQED’11)*, 2010, pp. 611–617.

[13] L. Wan and D. Chen, "Analysis of circuit dynamic behavior with timed ternary decision diagram," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 516–523.

[14] J. A. Kumar and S. Vasudevan, "Formal Probabilistic Timing Verification in RTL," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2013, vol. 32, no. 5, pp. 788–801.

[15] X. Wang and W. H. Robinson, "A Dual-Threshold Voltage Approach for Timing Speculation in CMOS Circuits," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 691–696.

[16] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical Timing Analysis : From Basic Principles to State of the Art," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 27, no. 4, pp. 589–607, 2008.

[17] L. Wan and D. Chen, "Analysis of Digital Circuit Dynamic Behavior With Timed Ternary Decision Diagrams for Better-Than-Worst-Case Design," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012, vol. 31, no. 5, pp. 662–675.

[18] L. Wan and D. Chen, "DynaTune : Circuit-Level Optimization for Timing Speculation Considering Dynamic Path Behavior," in *2009 International Conference on Computer-Aided Design (ICCAD), San Jose, CA*, 2009, pp. 172–179.

[19] L. Wan and D. Chen, "CCP: common case promotion for improved timing error resilience with energy efficiency," in *IEEE/ACM international symposium on Low Power Electronics and Design (ISLPED'12)*, 2012, p. 135.

[20] Y. Kuo, Y. Chang, and S. Chang, "Efficient Boolean Characteristic Function for Fast Timed ATPG," in *2006 IEEE/ACM International Conference on Computer Aided Design*, 2006, pp. 96–99.

[21] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary-decision diagrams," *ACM Comput. Surv.*, vol. 24, pp. 293–318, 1992.

[22] D. Ernest, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "Razor: circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, pp. 10–20, 2004.

[23] L. Benini, E. Macii, M. Poncino, and G. De Micheli, "Telescopic units: a new paradigm for performance optimization of VLSI designs," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 17, no. 3, pp. 220–232, Mar. 1998.

[24] R. E. Bryant, "Algorithms for Boolean Function Manipulation," vol. C, no. 8, 1986.

[25] T. Sasao, "Ternary decision diagrams. Survey," in *Proceedings 1997 27th International Symposium on Multiple- Valued Logic*, 1997, pp. 241–250.

[26] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack Redistribution for Graceful Degradation under Voltage Overscaling," in *Proceeding of ASP-DAC*, 2010, pp. 825–831.

[27] Berkely Logic Synthesis and Verification Group, “ABC: A system for sequential synthesis and verification.”

[28] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, “Razor : A Low-Power Pipeline Based on Circuit-Level Timing Speculation,” in *36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36)*, 2003, pp. 7–18.

[29] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, “RazorII : In Situ Error Detection and Correction for PVT and SER tolerance,” in *IEEE Journal of Solid-State Circuits*, 2009, vol. 44, no. 1, pp. 32–48.

[30] S. Das, S. Member, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, “A Self-Tuning DVS Processor Using Delay-Error Detection and Correction,” *J. Solid-State Circuits*, vol. 41, no. 4, pp. 792–804, 2006.

[31] P. Kocher, D. Genkin, D. Gruss, W. Haas, and M. Hambury, “Microarchitectural innovations: boosting microprocessor performance beyond semiconductor technology scaling,” *Proc. IEEE*, vol. 89, pp. 1560–1575, 2001.

[32] K. A. Bowman, J. W. Tschanz, N. S. Kim, J. C. Lee, C. B. Wilkerson, S.-L. L. Lu, T. Karnik, and V. K. De, “Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance,” in *IEEE Journal of Solid-State Circuits*, 2009, vol. 44, no. 1, pp. 49–63.

[33] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken, “TIMBER: Time borrowing and error relaying for online timing error resilience,” in *Design, Automation & Test in Europe Conference & Exhibition*, 2010, pp. 1554–1559.

[34] A. K. Uht, “Going Beyond Worst-Case Specs with TEAtime,” in *Computer*, 2004, vol. 37, no. 3, pp. 51–56.

[35] M. R. Choudhury and K. Mohanram, “Approximate logic circuits for low overhead, non-intrusive concurrent error detection,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE ’08)*, 2008, no. 0, pp. 903–908.

[36] M. R. Choudhury and K. Mohanram, “Low Cost Concurrent Error Masking Using Approximate Logic Circuits,” in *IEEE Transactions on computer-aided design of integrated circuits and systems*, 2013, vol. 32, no. 8, pp. 1163–1176.

[37] M. Pedram, “Power minimization in IC design: principles and applications,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 1, no. 1, pp. 3–56, 1996.

[38] V. Venkatachalam and M. Franz, “Power reduction techniques for microprocessor systems,” *ACM Comput. Surv.*, vol. 37, no. 3, pp. 195–237, 2005.

[39] L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, R. H. Dennard, and W. Haensch, “Practical strategies for power-efficient computing technologies,” *Proc. IEEE*, vol. 98, pp. 215–236, 2010.

[40] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, “Near-Threshold Computing: Reclaiming Moore’s Law Through Energy Efficient

Integrated Circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[41] S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, 3rd ed. Mc Graw Hill, 2003.

[42] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (NTV) design: opportunities and challenges," *49th ACM/EDAC/IEEE Des. Autom. Conf.*, pp. 1149–1154, 2012.

[43] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual: For System-on-Chip Design*. Springer Publishing Company, Incorporated, 2007.

[44] Y. Liu, R. Ye, F. Yuan, R. Kumar, and Q. Xu, "On Logic Synthesis for Timing Speculation," pp. 591–596, 2012.

[45] N. P. Carter, H. Naeimi, and D. S. Gardner, "Design techniques for cross-layer resilience," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2010, pp. 1023–1028.

[46] A. DeHon, H. M. Quinn, and N. P. Carter, "Vision for cross-layer optimization to address the dual challenges of energy and reliability," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2010, pp. 1017–1022.

[47] S. Mitra, K. Brelsford, and P. N. Sanda, "Cross-layer resilience challenges: Metrics and optimization," in *Design, Automation & Test in Europe Conference & Exhibition (DATE),* 2010, pp. 1029–1034.

[48] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "I-V power supply high-speed digital circuit technology with multi-threshold-voltage CMOS," *IEEE J. Solid-State Circuits*, vol. 30, pp. 847–854.

[49] Q. W. Q. Wang and S. B. K. Vrudhula, "Static power optimization of deep submicron CMOS circuits for dual V_T technology," *1998 IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. (IEEE Cat. No.98CB36287)*, pp. 490–496, 1998.

[50] L. Wei, Z. Chen, M. Johnson, and K. Roy, "Design and optimization of low voltage high performance dual threshold CMOS circuits," in *IEEE proceedings of Design Austomation Conference*, 1998, pp. 535–549.

[51] S. B. K. Vrudhula, "An investigation of power delay trade-offs for dual V/sub t/ CMOS circuits," *Proc. 1999 IEEE Int. Conf. Comput. Des. VLSI Comput. Process. (Cat. No.99CB37040)*, pp. 556–562, 1999.

[52] N. Jayakumar and S. P. Khatri, "A Predictably Low-Leakage ASIC Design Style," vol. 15, no. 3, pp. 276–285, 2007.

[53] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *ArXiv e-prints*, 2018.

[54] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre Attacks: Exploiting Speculative

Execution," 2018.

[55] F. Brglez, "A neutral netlist of 10 combinational benchmark circuits and a target translation in FORTRAN," *IEEE Int. Symp. Circuits Syst.*, 1985.

[56] M. Hansen, H. Yalcin, and J. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," *IEEE Des. Test Comput.*, vol. 16, pp. 72–80, 1999.

[57] "Synopsys. Synopsys University Program. Available: http://www.synopsys.com/Community/UniversityProgram/Pages/default.aspx." .

[58] K. E. Murray, A. Suardi, V. Betz and G. Constantinides, "Calculated Risks: Quantifying Timing Error Probability with Extended Static Timing Analysis," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.