REVAMPING TIMING ERROR RESILIENCE TO TACKLE CHOKE POINTS AT NTC

by

Aatreyi Bal

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

Sanghamitra Roy, Ph.D. Major Professor	Koushik Chakraborty, Ph.D. Committee Member
Jacob Gunther, Ph.D. Committee Member	Reyhan Baktur, Ph.D. Committee Member
Vicki Allan, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

ABSTRACT

Revamping Timing Error Resilience to Tackle Choke Points at NTC

by

Aatreyi Bal, Doctor of Philosophy

Utah State University, 2019

Major Professor: Sanghamitra Roy, Ph.D.
Department: Electrical and Computer Engineering

Process variation (PV) is a conspicuous predicament for sub-micron VLSI circuits. Near Threshold Computing (NTC) systems have been inherently plagued with heightened PV sensitivity. *Choke points* are an intriguing manifestation of this PV sensitivity. Choke points are a severe reliability concern in post-silicon NTC systems. They are capable of dominating the choice of critical paths in a fabricated chip, as well as, shortening an already short delay path due to reduced gate delays. This dissertation investigates the different aspects of choke points and their non-trivial impacts on the system reliability. It is also demonstrated that blindly adopting conventional timing error mitigation techniques into NTC will fail to efficiently tackle choke errors. Consequently, two dynamic and adaptive techniques are suggested which not only mitigate the choke errors, but also ensure to maintain the energy efficiency of NTC systems. The proposed techniques show significant improvement over popular super threshold computing (STC) techniques in terms of both performance and energy efficiency, at the cost of minimal hardware overheads.

(73 pages)

PUBLIC ABSTRACT

Revamping Timing Error Resilience to Tackle Choke Points at NTC

Aatreyi Bal

The growing market of portable devices and smart wearables has contributed to innovation and development of systems with longer battery-life. While Near Threshold Computing (NTC) systems address the need for longer battery-life, they have certain limitations. NTC systems are prone to be significantly affected by variations in the fabrication process, commonly called process variation (PV). This dissertation explores an intriguing effect of PV, called choke points. *Choke points* are especially important due to their multifarious influence on the functional correctness of an NTC system. This work shows why novel research is required in this direction and proposes two techniques to resolve the problems created by choke points, while maintaining the reduced power needs.

To Mummy, Baba and Somdeb.

ACKNOWLEDGMENTS

The completion of this Doctoral degree could not have been possible without the help, encouragement and support from a lot of people. Firstly, I would like to thank my advisor Dr. Sanghamitra Roy, and my co-advisor Dr. Koushik Chakraborty for accepting me as a part of USU Bridge Lab. Their constant encouragement, technical guidance and constructive criticism were my assets in my strive towards meaningful research. Beyond advising, they were the mentors who ensured that I have a fun and healthy PhD life. Secondly, I would like to thank the Research and Graduate School of the Utah State University for majorly funding my PhD research in the form of Presidential Doctoral Research Fellowship (PDRF). The invaluable opportunities and help provided by Dr. Scott Bates and the tireless effort of Athena Dupont need special mention. Most importantly, I want to thank my respectable committee members; Dr. Jacob Gunther for his wise words of encouragement, Dr. Reyhan Baktur for her insightful comments, and Dr. Vicki Allan for her motivating presence. I would like to specially thank Tricia Brandenburg, Kathy Phippen and Diane Buist, for their ready assistance with all the paperworks and my incessant queries.

I would like to take this opportunity to thank a few more people who have enriched my life above and beyond the professional field. Anusna, for being the first person to welcome me in this new city and my first friend here. Without her, the initial days would have been a confusing struggle. Bidisha di, for lighting up my mundane life with her occassional presence. And of course, Soodeh, who has been my closest friend and confidante over past two years. I would like to thank my lab members, who have grown to be my family in this country. Rajesh, who was the first person from Bridge lab I had contacted, and the person who has continuously inspired me with his calm, funny and dignified personality. Prabal, who has always surprised me with his depth of knowledge (and vocabulary) in diverse aspects of research and life. Chidham, who was my one-stop-solution for technical details and meaningless chitchats. Asmita, who made my life in lab easier and fun. Pramesh, who always managed to make me smile with his sheer presence. Sourav, who was the sole target of my childish jokes and pranks. Tahmoures, who accepted my scolding with a smile and still managed to not hate me. Finally, Shamik, for partnering and helping with my first paper in this lab and Hu, for his help and advice. I would also thank all my school friends and Soham, for always being by my side, no matter what.

My journey so gfar would have been a distant dream without the sacrifice and blessings of my parents, Mr. Pradip Kumar Bal and Mrs. Babila Bal. I drew my inspiration from Dr. Subhajit Datta and Dr. Reshmi Das, who always knew what I can be, even when I doubted. In addition to my parents, it was the blessings and encouragement of my family, Mr. Priyabrata Mittra, Mrs. Sharmila Mittra, Dr. Kaustuv Mittra and Dr. Kanika Sharma Mittra, that has kept me going. I must mention a very special person, Dr. Basudeb Biswas, whose unconditional love and blessings have enriched my life in the very small time that I have got to know him.

Lastly, I am ever-so thankful to my loving husband, Mr. Somdeb Mittra, for being my all-round support system in this hurdle race called graduate life.

Aatreyi Bal

CONTENTS

LIST OF FIGURES

ACRONYMS

OWM	Operand Width Marker
DCS	Dynamic Choke Sensing
CC	Choke Controller
CSLT	Choke Sensor Lookup Table
CDC	Choke Detection Controller
TDC	Transition Detector and Counter
CCR	Choke Clearance Register
EID	Error Identifier
CET	Choke Error Table
CDL	Choke Delay Level
CGL	Choke Gate Level
NTC	Near Threshold Computing
STC	Super Threshold Computing
PV	Process Variation
DCS-ICSLT	DCS-Independent Choke Sensor Lookup Table
DCS-ACSLT	DCS-Associative Choke Sensor Lookup Table
PC	Program Counter
De	Decode pipestage
Ex	Execute pipestage
WB	Writeback pipestage
NTV	Near Threshold Voltage
PTM	Predictive Technology Model
STA	Statistical Timing Analysis
RTL	Register Transfer Level
ALU	Arithmetic Logic Unit
EDAC	Error Detection And Correction

CHAPTER 1
INTRODUCTION

The evolution of *Internet of things* (IoT), over the past decade, has taken the form of an industrial revolution, both economically and technologically. The IoT boom has facilitated the growth of smart gadgets like smart sensors, smart wearables, smart lights, and so forth. The exponential growth curve of the IoT ecosystem necessitates the emergence of low power platforms and devices. Consequently, *Near-Threshold Computing* (NTC) has gained popularity as a promising design paradigm for such low-power energy-efficient computing systems [1, 2]. NTC systems have a supply voltage marginally higher than the threshold voltage of the constituent devices. The supply voltage scaling results in multiple orders of magnitude improvement in energy efficiency. However, with the advent of Edge Computing[1], energy efficiency is no more the only concern; reliability has emerged as a major requirement for IoT devices [3].

NTC fails to bridge the gap between reliability and energy efficiency [1, 4]. NTC systems experience higher process variation (PV) sensitivity, compared to conventional Super-Threshold Computing (STC) systems [5]. This PV sensitivity is commonly manifested in the form of gate delay variations. PV induced gate delay variations can be as large as 20× compared to their nominal values, in NTC systems [6]. The drastic gate delay variations, in turn, cause significant path delay variations. The distinct reliability concerns posed by these PV induced delay variations at NTC, cannot be mitigated simply by adopting the traditional error mitigation techniques designed at STC. [7]. This dissertation aims to address a key research question in this context: *How can these unique reliability challenges be tackled at NTC, while still continuing to harvest the energy efficiency benefits offered by the voltage scaling?*

[1]Technologies enabling data processing/computation at the edge of the network.

1.1 Choke Points: A Unique Challenge

In the quest for unique reliability challenges at NTC, *Choke Points* have emerged as a pivotal player [8]. A *choke point* comprises a single or a small group of PV affected gate(s) that practically dominates the delay of the entire path in which it occurs. Though choke points can occur anywhere in a circuit, their impacts are observed only if the corresponding paths are sensitized [2]. However, being an artifact of the fabrication process, choke points cannot be estimated at design time. This work explores the the concept and various manifestations of choke points and strives to mitigate their impact on the reliability of NTC systems.

1.2 Contributions of This Dissertation

The research works associated with this dissertation have been partially published in various conference and journals, including 2017 IEEE/ACM Design, Automation and Test in Europe (DATE), 2018 IEEE/ACM Design, Automation and Test in Europe (DATE) (nominated for Best Paper Award), 2019 IEEE/ACM Design, Automation and Test in Europe (DATE), 2018 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), and 2018 IEEE Transactions on Very Large Scale Integration Systems (TVLSI) (January and November issues). Publications stemming from this dissertation are listed as follows:

1.2.1 Conference Papers

- Aatreyi Bal, Sanghamitra Roy, and Koushik Chakraborty, Trident: A Comprehensive Timing Error Resilient Technique against Choke Points at NTC, Proceedings of the IEEE/ACM Design, Automation and Test in Europe (DATE), March 2018 (Received Nomination for Best Paper Award).
- Aatreyi Bal, Shamik Saha, Sanghamitra Roy, Koushik Chakraborty, Revamping timing error resilience to tackle choke points at NTC systems, Proceedings of the IEEE/ACM

[2]A path is sensitized when the applied input changes the output state of the path.

Design, Automation and Test in Europe (DATE), March 2017.

- Sourav sanyal, Prabal Basu, Aatreyi Bal, Sanghamitra Roy and Koushik Chakraborty, Predicting Critical Warps in Near-Threshold GPGPU Applications using a Dynamic Choke Point Analysis, IEEE/ACM Design, Automation and Test in Europe (DATE) 2019 (accepted).

- Tahmoures Shabanian, Aatreyi Bal, Prabal Basu, Koushik Chakraborty and Sanghamitra Roy, ACE-GPU: Tackling Choke Point Induced Performance Bottlenecks in a Near-Threshold Computing GPU, ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2018.

1.2.2 Journal Papers

- Aatreyi Bal, Sanghamitra Roy and Koushik Chakraborty, Trident: Comprehensive Choke Error Mitigation in NTC Systems,IEEE Transactions on Very Large Scale Integration Systems (TVLSI), November 2018.

- Aatreyi Bal, Shamik Saha, Sanghamitra Roy, Koushik Chakraborty, Dynamic Choke Sensing for Timing Error Resilience in NTC Systems, IEEE Transactions on VLSI Systems (TVLSI), January 2018.

CHAPTER 2

LITERATURE REVIEW

This chapter lays out a comprehensive literature survey on existing works that are closely associated with the work presented in this dissertation. The problem statement of this dissertation is focussed on addressing a pivotal timing error source in NTC systems, while maintaining their inherent energy-efficiency. Consequently, the contemporary works related to this dissertation can be broadly classified into two categories: opportunities and challenges in NTC system design (Section 2.1) and state-of-the-art timing error detection and correction (EDAC) techniques (Section 2.2). Works pertaining to each of the category are discussed in detail next. Finally, Section 2.3 prominently outlines the contributions of this dissertation in the research on reliable NTC system designs.

2.1 Opportunities and Challenges at NTC

The promises offered by the NTC design paradigm, though manifold, have been diminished by various intrinsic vulnerabilities. However, researchers have historically focussed on harnessing the benefits of NTC systems, with minimal attention to their fragility. While Section 2.1.1 lists the works that highlight the promising design paradigm shift offered by NTC, Section 2.1.2 focuses on the handful of recent works aimed at identifying and alleviating the reliability concerns in NTC systems.

2.1.1 Exploring the Opportunities in NTC System Design

- **Dreslinski et al. [1]:** This work introduces the multi-dimensional opportunities and barriers of NTC system design. NTC domain offers energy reduction on the order of 10× at the cost of approximately 10× performance degradation. Further, the effects of process and environmental variations induce about 5× performance variation. Frequent functional failures also limit the potential of NTC system design. The authors present a detailed explanation of factors inhibiting the widespread acceptance of NTC in the domain of system design.

- **Kaul et al.** [9]**:** Besides advocating the benefits of near threshold voltage (NTV) operation, this work proposes several design and optimization techniques for gates, registers, latches etc., to ensure reliability of the system. The authors strongly champion the large scale use of NTV in future computing systems.

- **Markovic et al.** [10]**:** The authors of this paper propose an energy-delay framework for efficient NTC systems. This work demonstrates that, in comparison to the conventional STC techniques like gate-sizing, voltage modulation is a better trade-off knob for NTC systems. This paper also shows that the use of time-multiplexing around the minimum energy point results in both lower area and energy without significant performance penalty.

- **EnergySmart** [11]**:** This paper proposes a manycore organization with only a single supply voltage (V_{dd}) and multiple frequency domains. The authors show that multiple voltage domains are inefficient for manycore NTC chips. The authors pair their approach with fine-grained dynamic voltage and frequency scaling (DVFS) to enhance its competency. This work further introduces a core-job assignment for the EnergySmart architecture.

- **Centip3De** [12]**:** This paper presents a large-scale 3D chip multiprocessor with a cluster-based near-threshold computing (NTC) architecture. Each cluster consists of 4 cores and accesses a single cache that is 4× larger than the conventional independent caches. The larger caches are operated at higher voltage and frequency to cater to the needs of the cores. In order to speed-up the serial portions of a parallelized application, this work introduces per-core DVFS, in addition to architectural boosting techniques.

- **Dogan** [13]**:** This paper presents a practical use case of NTC systems. The authors propose a near-threshold multi-core architecture, capable of executing biomedical applications, with multiple instruction and data memories. This architecture also includes broadcasting mechanisms for the data and instruction memories to optimize system energy consumption by tailoring memory sharing to the target application.

2.1.2 Design Challenges at NTC

- **VARIUS-NTV [5]:** This work presents a microarchitectural model for the process variations at NTC. This paper builds on the VARIUS model [14] for STC systems. Besides modelling the gate delays at NTC, this paper demonstrates a detailed modelling of SRAM suitable for NTC systems. The authors model how variation affects the frequency attained and power consumed by cores and memories in an NTC manycore, and the timing and stability faults in SRAM cells at NTC.

- **VARIUS-TC [15]:** This work focuses on a modular architecture-level model of parametric variation in thin-channel switches, like FinFETs. This paper decouples the architecture-level model from the device and circuit-level models, through proper abstraction.

- **Variation-aware Synthesis [16]:** This paper advocates a process-variation aware design phase for NTC systems. Consequently, the authors propose a process-variation aware circuit synthesis. The proposed synthesis technique is iterative in nature. Initially the circuit is synthesized and statistical static timing analysis (SSTA) is performed to estimate the impact of process variation. Next, the variation information in provided to the synthesis tool to re-synthesize the circuit considering the timing constraints.

- **Kim et al. [17]:** This paper proposes an in-situ error detection and correction technique (EDAC) for resilient ultra-low voltage systems. The technique is voltage-scalable and incurs low overhead in terms of area/energy/throughput. The proposed technique performs a sparse error detection strategy. The error detection and correction is done within one cycle, without stalling the pipeline.

- **Golanbari et al. [18]:** This work specifically addresses the increased count of hold time violations in NTC systems. The authors demonstrate that conventional state-of-the-art hold time fixing techniques are not efficient at near threshold voltages. This paper proposes a SSTA based iterative technique to fix the hold time violations at NTC. The proposed technique uses transmission-gate based buffers to tackle the minimum path delay constraint.

- **De [8]:** This paper first introduces the term choke point. The author presents the challenges of designing manycore system-on-chip(SoC) at NTC. The paper presents a detailed energy analysis of NTC systems, in comparison to their STC counterparts.

2.2 State-of-the-art EDAC Techniques

In contrast to the few timing error resilience models at NTC, STC has a significant body of work presenting state-of-the-art EDAC techniques. EDAC techniques can be classified as either *reactive* — triggered only after a timing error occurs — or *proactive* — speculates the occurrence of imminent timing errors. Existing works in each of these classes of techniques are discussed in Sections 2.2.1 and 2.2.2, respectively.

2.2.1 Reactive Techniques

- **Razor [19]:** This paper proposes a popular EDAC technique for STC systems. The technique uses double sampling flip-flops to detect the errors. Pipeline flush and instruction replay at a reduced frequency are used to correct them. Razor uses buffer insertion to avoid minimum timing violations. The error detection and correction penalty cycles is equal to the number of stages in the pipeline.

- **Online Clock-Skew Tuning [20]:** This paper is based on repeated clock-skew tuning to avoid timing errors. This technique divides the circuit into blocks and observes the errors in each of the blocks for a given short interval of time. While observing, the EDAC method followed is same as Razor. If the error frequency crosses a certain threshold, the clock skew for that block is tuned such as to allow extra clock period to complete execution without errors.

- **HFG [21]:** This paper proposes a high-level model for timing error rate (TER). In addition, this paper proposes a technique, called Hierarchically Focused Guardbanding (HFG), that relies on in-situ sensors to monitor the effects of process, voltage, temperature and aging (PVTA) in a system. Subsequently, HFG modulates the timing guardbands to eliminate the resulting errors.

2.2.2 Proactive Techniques

- **Lak et al. [22]:** This technique uses sensors to monitor device aging in circuit paths. The monitoring helps in identifying evolving critical paths and predicting imminent timing errors, which are then avoided by adaptive clock tuning mechanism.
- **Xin et al. [23]:** This work uses pronounced locality in instruction-level error rates due to value locality and data dependences to predict timing errors in pipeline processors. This instruction-level prediction and error-padding technique significantly reduces the penalty incurred due to repeated error detection and correction.
- **Roy et al. [24]:** This paper proposes a program counter (PC) based error prediction technique. Besides considering history of timing violations caused by an instruction, the proposed technique also considers the operating conditions to improve the prediction accuracy.

2.3 Choke Error Resiliency at NTC

Apart from [8], the works presented in this dissertation are the first to have investigated choke points. The works have explored the impacts of choke points across the STC and NTC domain. The results corroborate that choke points are a greater hurdle in NTC systems, compared to STC systems. Furthermore, the randomness in the occurrence and impacts of choke points are only manifested when the corresponding paths are sensitized.

Consequently, tackling choke points requires dynamically adaptable techniques. This dissertation demonstrates the inefficacy of state-of-the-art techniques to handle this situation, as well as, proposes techniques to comprehensively deal with the choke errors. While the first technique presented in this dissertation, *Dynamic Choke Sensing*, only focuses on maximum timing violations, the second technique, *Trident*, addresses a broader range of errors. Besides errors induced by maximum timing violations, this technique addresses errors induced by minimum timing violations and the errors caused by the interplay of maximum and minimum timing violations. This dissertation also introduces the concept of *choke buffers*, i.e., buffers acting as choke points. This concept diminishes the efficacy of buffer insertion technique, commonly used to avoid minimum timing violations in short delay paths. Hence, the works presented in this dissertation tread largely uncharted territories in the realm of reliable NTC systems.

CHAPTER 3

DYNAMIC CHOKE SENSING FOR TIMING ERROR RESILIENCE IN NTC SYSTEMS

3.1 Background and Contributions of This Work

Near-Threshold Computing (NTC) has emerged as one of the promising directions in the pursuit of improving the energy-efficiency of computer designs [1, 25, 26] —a growing concern in the current geopolitical landscape. A device operating at NTC sets its supply voltage close to its threshold voltage, while still maintaining a positive difference between them. Consequently, the energy consumption is dramatically reduced, both due to the quadratic impact of supply voltage reduction, as well as, a linear reduction from the operating frequency. This reduction eliminates the risk of hitting the power wall in case of multicore processors. The multicore systems operating at NTC aim at recovering the performance loss, due to reduced frequency, by enhancing parallel application across the cores [27]. Therefore, NTC plays a pivotal role in mitigating dark silicon [28]. Further, with the advent of low-power wearables, NTC has evolved as a prominent design paradigm [29].

The tremendous (multiple orders of magnitude) improvement in energy efficiency at NTC [9], comes with its own set of challenges. Beside 10× or more performance degradation, NTC suffers from exacerbated process variation (PV) sensitivity [5,7]. PV at NTC can vary the gate delays by as large as 20× of their nominal values [6]. Due to this massive delay variation, NTC circuits have a substantially higher reliability threat from *choke points*—a small set of PV affected gates that can dominate the selection of critical paths in the post-silicon circuit—than their Super-Threshold Computing (STC) counterparts [8].

In this paper, an extensive circuit-architectural analysis is used to illustrate how choke points are formed and their resulting impact on conventional STC circuits and rapidly evolving NTC circuits, respectively. It is observed that a choke point can be formed with as small as 0.17% of the total gates in an NTC ALU—a circuit with a large logic depth—causing a delay degradation of 23.7%. These intriguing characteristics can substantially degrade the critical path delay at NTC, while also radically altering the composition of critical paths in a fabricated circuit by massively degrading the delay in short paths. In addition, choke points cannot be estimated pre-fabrication. A batch of identical chips may have a large variation in the distribution of choke points, post silicon. Thus, existing physical design techniques are rendered inefficient in mitigating this problem. To overcome these limitations, a low overhead and dynamically adaptive timing error mitigation technique is proposed, called Dynamic Choke Sensing (DCS).

The main contributions of this work are as follows:

- Process variation induced gate-delay deviation at NTC and its role as a crucial source of timing errors are the major focal points of this work. Choke points are established as a significant outcome of this delay variance (Section 3.2).

- A low overhead dynamic timing error prediction and mitigation technique, called Dynamic Choke Sensing (DCS), is proposed. This technique performs an early choke point detection and, thereby, uses the knowledge to avoid recurrent timing errors. As a result, penalty cycles incurred to recover from timing errors are reduced (Section 4.3).

- Two variants of the scheme are proposed, categorised on the basis of the structure of the lookup table. A comparative study of the proposed schemes with other contemporary schemes are presented later in the work.

- It is demonstrated that the schemes provide 30%-55% improvement in performance and 60%-73% improvement in energy efficiency as compared to representative timing error recovery technique, Razor [19] (Section 4.5). The area, wire-length and power overheads of DCS schemes are 0.23%-0.48%, 0.77%-0.85% and 0.85%-1.2%, respectively.

3.2 Motivation

In this section, it is demonstrated that circuits operating at NTC have a substantially higher chance of manifesting choke points than their STC counterparts. An extensive circuit-architectural analysis is also presented to illustrate how a small number of PV affected gates at NTC can serve as choke points, potentially transforming a shorter delay path into a critical path, after chip fabrication. Finally, the unique challenges in robust NTC circuit design, engendered by choke points, are presented. In Section 3.2.1, *choke points* are defined and their potency is discussed. Sections 3.2.2, 3.2.3 and 3.2.4 present the methodology, results and significance of this study, respectively.

3.2.1 Choke Points

A choke point comprises a single or a small group of PV affected gate(s) that can transform a shorter delay path into a critical path (Choke Path), when sensitized[1]. Further, choke points can also create critical paths with substantially higher delays than the nominal. Figure 3.1 illustrates the concept of a choke point in a small circuit. P1 is the nominal critical path. A post-silicon instance of this circuit, however, suffers a high delay variation due to

[1]A path is sensitized when the applied input changes the output state of the path.

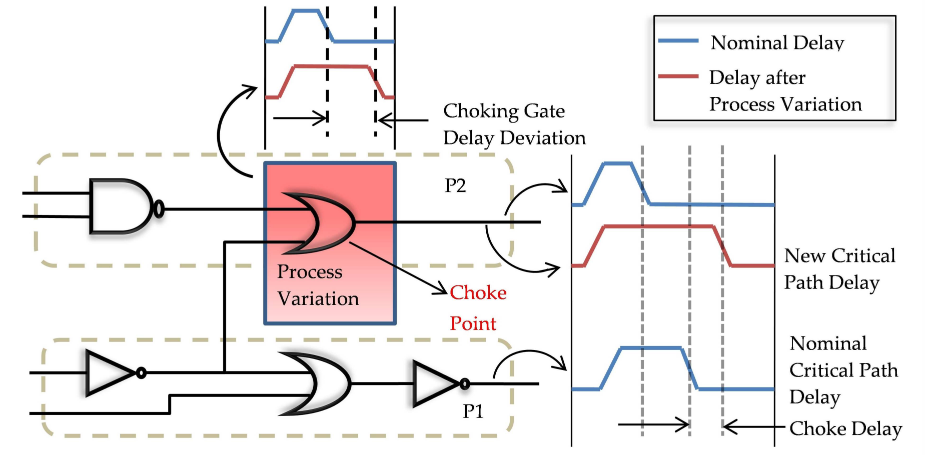
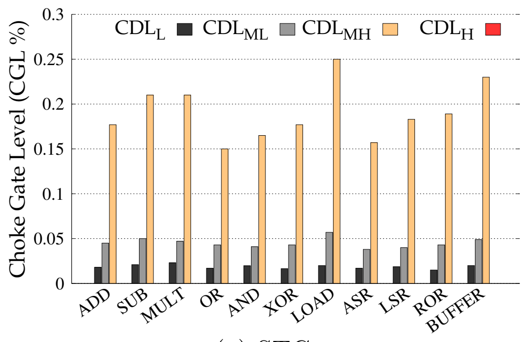
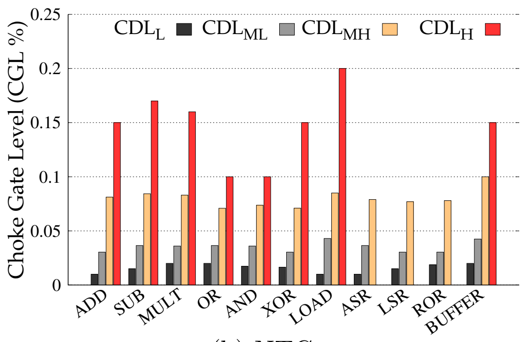

Nominal Delay
Delay after Process Variation
Choking Gate Delay Deviation
P2
Process Variation
Choke Point
New Critical Path Delay
Nominal Critical Path Delay
P1
Choke Delay

Fig. 3.1: P1 is the nominal critical path and P2 is a non-critical path pre-fabrication. P2 becomes the new critical path post-fabrication owing to the increased gate delay of the choke point.



(a) STC



(b) NTC

Fig. 3.2: Choke Gate Level(CGL) plot of each ALU operation for four distinct categories of Choke Delay Level(CDL). Note that the Y-axis scales upto 0.25% and 0.2% at STC and NTC, respectively.

the OR gate in the path P2. The OR gate serves as a choke point, transforming P2—a short delay path—as the new critical path.

Though choke points can be formed anywhere in a circuit, their effects are observed only in sensitized paths. Typically, the distribution of sensitized paths in a circuit depends on the instructions executed on the circuit, as well as, the inputs to those instructions.

Potency of a Choke Point

Two key parameters are defined—Choke Delay Level (CDL) and Choke Gate Level (CGL)—to quantify the potency of choke points. CDL is the amount of additional delay introduced by the choke point to create the new critical path, overshooting the nominal critical path delay. It is expressed as a percentage of the nominal critical path delay. CGL is the number of gates forming the choke point, expressed as a percentage of the total gates in the whole circuit. A low CGL signifies a highly potent choke point, where, a small percentage of gates in the circuit can transform the critical path. Similarly, a high CDL also indicates high potency of a choke point.

Threat of a Choke Point

Analyzing the effects of choke points in the sensitized paths is of paramount importance for the reliable operation of a NTC system. To understand the extent of threat presented by choke points in modern processors, a few key research questions are posed. *How can the* *significance of choke points be compared in STC and NTC circuits? What is the likelihood of choke points in transforming critical paths in popular processor pipelines while running real applications?* To answer these compelling questions, a rigorous cross-layer methodology is employed, outlined next.

3.2.2 Methodology

Analyzing sensitized choke points in STC and NTC presents a methodological challenge. As instructions are executed in a circuit component, they sensitize different paths, and therefore observe different logic computation delays. Further, the sensitized path in a circuit depends on two consecutive instructions [30]. For this analysis, the PV in the logic gates at STC and NTC are modeled on the basis of VARIUS [14] and VARIUS-NTV [5], respectively. The supply voltages are set at 0.8V and 0.45V for STC and NTC, respectively. Next, a 64-bit ALU is synthesized using a 15nm FinFET library from NanGate [31]. The PV-affected logic gate models are used in the in-house Statistical Timing Analysis (STA) tool to study the cycle accurate delay timings of all the sensitized paths for 11 different arithmetic and logic operations. The operands are chosen to cover a typical range seen in real applications. The sensitized path delay distribution for different combinations of operations and operands are extensively studied to analyze potential choke paths. A detailed description of methodology is presented in Section 4.4.

3.2.3 Results

Figure 3.2a depicts the correlation between CDL and CGL for each operation of the ALU at STC. The data for four categories of CDL are presented: CDL_L or CDL-Low (0-5%), CDL_ML or CDL-Medium Low (>5%-10%), CDL_MH or CDL-Medium High (>10%-20%) and CDL_H or CDL-High (>20%-30%). Choke points are found to be formed typically in the range of 0 to 12% CDL, at STC. This trend can be attributed to the fact that individual gate delay deviation at STC is not large enough to surpass a CDL larger than 12%, even when all the gates in the choke path are affected by PV.

Figure 3.2b explores the same correlation at NTC. It is observed that choke paths can be formed at higher CDL, with substantially smaller CGL, at NTC. For example, in case of LOAD operation, a CGL of only 0.2% is sufficient to surpass a CDL of 27.45% while for AND, only 0.1% CGL can exceed a CDL of 23%. However, the choke point sensitization in the higher range $\mathrm{CDL_H}$, varies across operations. In a few operations (like ASR, LSR and ROR), choke points fail to create new critical paths with higher CDL. The reason being, in the higher categories of CDL, typically the path with the maximum logic depth forms the critical path. Thus, other sensitized paths, with lesser logic depth, fail to emerge as potential choke paths in $\mathrm{CDL_H}$ category, for these operations. On a deeper analysis, it is found that choke point sensitization depends on the following architectural factors:

- **Nature of Operations**: The computation complexity involved in an operation significantly affects the path sensitization at a given cycle. The greater the number of intermediate calculation steps involved, more is the number of sensitized gates in the circuit. Consequently, the probability of most of the PV affected gates appearing in the sensitized paths, and thereby the potency of choke point creation, increases. This characteristic is clearly depicted by the ADD and BUFFER operations. Since ADD sensitizes more paths while calculating sum and carry, chances of sensitizing more PV affected gates in the process is high. BUFFER on the other hand simply passes the input to the output after one clock cycle. Therefore, in every category of CDL, ADD records lower CGL than BUFFER, depicting its higher potency in creating choke paths.

- **Significant Width of Operands**: The significant width of the operands of an operation—the number of set bits in the operand—affects the probability of choke point formation. If the significant width is smaller than half the total width of operand, determined by the Instruction Set Architecture (ISA), it is termed ‘low’; otherwise it is considered ‘high’. Higher the significant width, more gates are likely to be sensitized across the paths, and greater is the possibility of the PV affected gates occurring in those paths. Therefore, higher significant operand widths have greater potency to

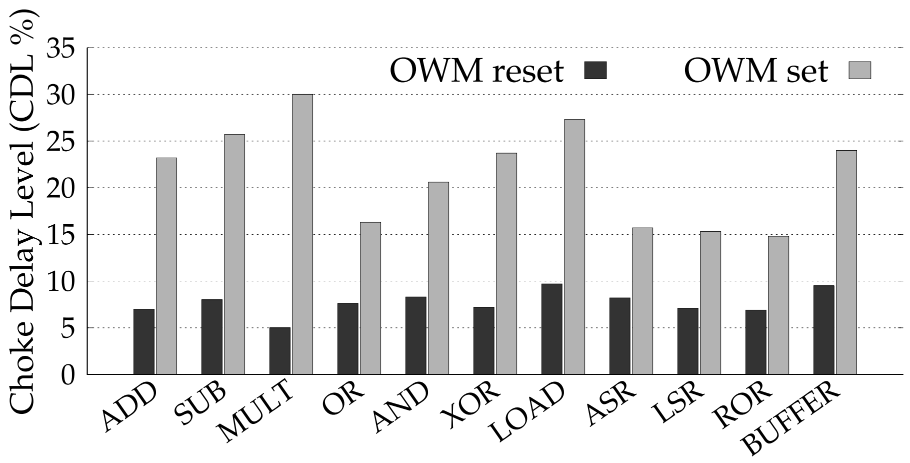


Fig. 3.3: Choke Delay Level (CDL) variation with Operand Width Marker (OWM) for each operation at NTC. Choke paths with higher CDL can be created when OWM is set, signifying higher significant width of either or both operands of an operation. Reset value of OWM signifies both the operands have low significant width and is less potent in creating choke paths.

create choke paths, even in the higher categories of CDL. The metric Operand Width Marker (OWM) is used to denote if either of the operands in an operation has high significant width. If none of the operands has high significant width, OWM has reset value; else, it has set value. The potency of significant operand width in forming choke paths is depicted in Figure 3.3.

- **Sequence of Instructions**: The occurrence of timing errors is dependent on the sensitizing vector as well as the initializing vector [30]. The sensitizing vector is the errant instruction that reveals the timing fault. The initializing vector is the instruction executed in the previous cycle that determines the initial states of the paths and nodes. Figure 3.4 displays the errant and error-free occurrence percentages of few instructions in the *vortex* benchmark. While NOR instruction causes timing error every time it is executed, ADDIU shows error-free execution for about 50% of its occurrences.

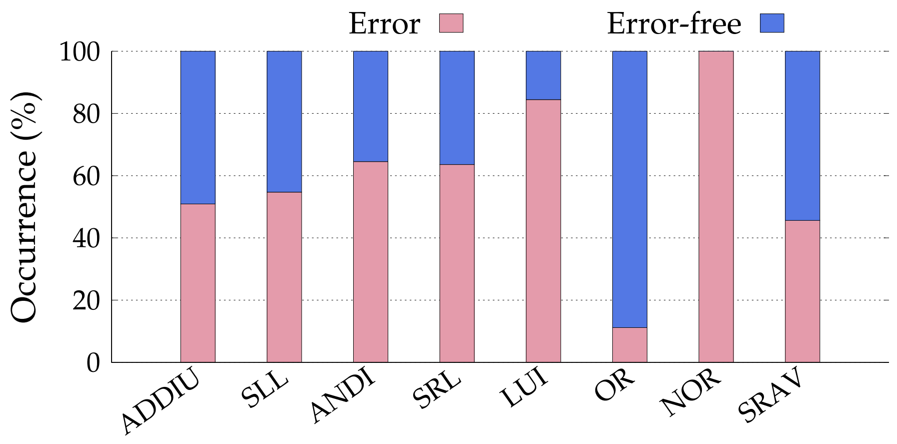


Fig. 3.4: Comparison of errant and error-free occurrence percentages of few instructions in *vortex*.

3.2.4 Significance

The observations show that the heightened effect of PV on gate delay at NTC can massively degrade a short delay path into a choke path or substantially degrade the critical path delay in a fabricated circuit. In addition, choke points—an artifact of PV and NTC operation—cannot be estimated pre-fabrication. In fact, a batch of identical chips may have a large variation in choke paths, post silicon. This intriguing characteristic can render conventional physical design techniques of timing error mitigation, like gate-sizing and multiple threshold voltage (V_{th}) assignment, inefficient. Likewise, existing techniques like hierarchical guardbanding [21] or timing speculation can suffer from substantial energy efficiency loss in mitigating timing errors from choke paths. Then, a key question is *how can a low overhead hardware be designed inside every chip that can learn and adapt to its own choke paths?* The next section discusses the proposed scheme, Dynamic Choke Sensing (DCS), to mitigate timing errors from choke points at NTC.

3.3 DCS Design

In this section, a robust technique, Dynamic Choke Sensing (DCS), is presented which exploits the factors discussed in Section 3.2.3, to mitigate timing errors. Choke points vary from chip-to-chip within the same design, due to variance in the degree of PV; but they comprise a permanent characteristic of a particular chip instance. Timing violations caused by choke points emerge as an inherent property of the chip. Though, newer timing violations may arise or existing violations may magnify due to aging, yet, an existing choke point will continue to cause timing violations for the entire lifetime of the chip. The proposed technique adaptively mitigates these dynamic timing errors. In Section 3.3.1 a brief overview of the technique is presented. In Section 3.3.2, the error tagging mechanism is discussed and in Section 3.3.3, the two variants of the DCS scheme are presented. Sections 3.3.4 and 3.3.5 highlight the methodology of the techniques and the error handling mechanisms, respectively.

3.3.1 DCS Overview

Figure 3.5 portrays an overview of the components and flow of the DCS technique. DCS focuses on sensing errors and avoiding their recurrent occurrences. The two major components facilitating this technique are the Choke Controller and the Choke Sensor Lookup Table (CSLT). The CSLT serves as a record of the unique timing error instances. The Choke Controller performs the pipeline flush and instruction replay, when an error is detected for the first time, and inserts the erroneous opcode into the CSLT. When the same errant opcode is identified in the decode stage of the pipeline, the Choke Controller avoids the error by inserting a stall cycle. It is assumed that, even in the presence of worst-case timing error, an instruction finishes execution in maximum two cycles. Thus, these two components perform in unison to reduce the penalty cycles for repeated timing error recovery, thereby improving the overall performance of the system.

3.3.2 Error Tagging

The DCS scheme implements a unique method to tag error instances. Instead of using the program counter values, the errant instructions are tagged by their opcodes. Further, in Section 3.2.3, it was observed that timing errors are a function of the operand width and the previous cycle instruction. So, a single error instance has a four-part tag: (a) errant instruction opcode, (b) errant instruction OWM, (c) previous cycle opcode and (d) previous

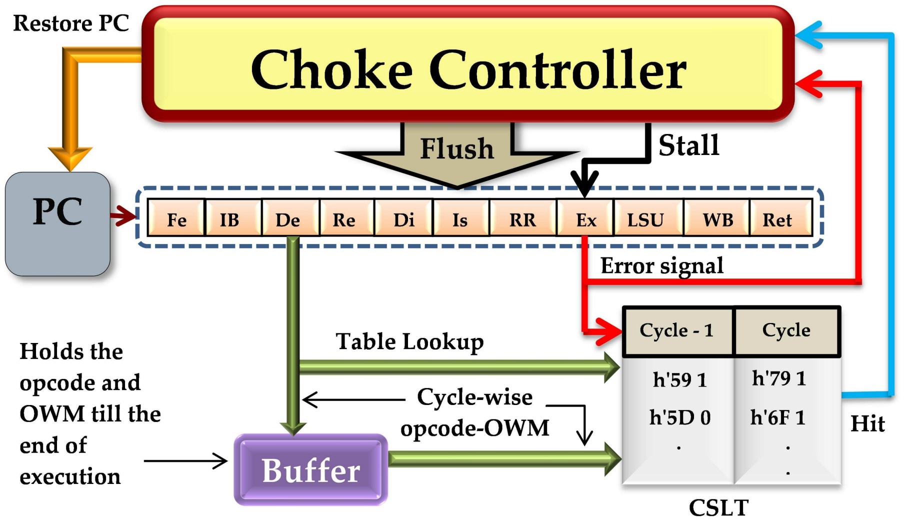


Fig. 3.5: Every cycle, the decoded opcode is looked up in the table. If there is a match, in both current and previous cycle opcode-OWM, the Hit signal is set high. If there is no match and an error is detected in the execute stage (Ex), the Error signal is set high and the 8-bit opcode and 1-bit OWM for the errant cycle and previous cycle from the buffer are latched to the CSLT. The Choke Controller regularly checks both the Hit and Error signal. If the Hit is high, it inserts a stall cycle in Ex-stage for corresponding opcode. If Error signal is high, pipeline flush and instruction replay are initiated by the controller.

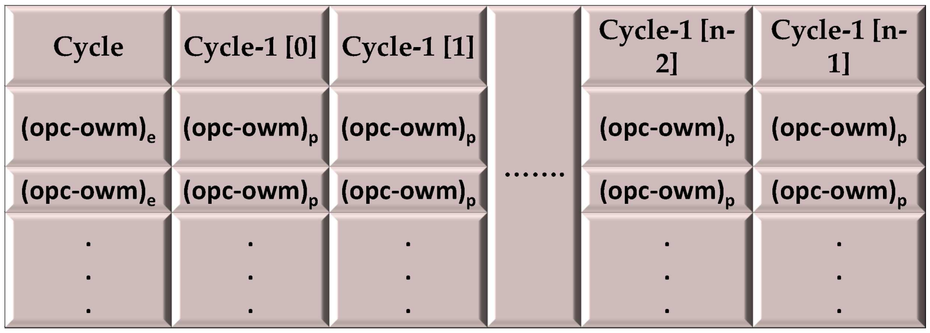
cycle OWM. These tags are then stored in the CSLT for future reference of probable timing errors. The opcode-OWM sequence characterizes the changes in the output states of the path, over this one cycle period, which trigger the timing error. Earlier works on predictive schemes, such as [24, 32], have relied on simple program counter (PC) tags to record, and refer to, error instances. However, the unique tag, used for DCS, allows us to monitor the timing error instances at a finer granularity, and thereby, more precisely.

3.3.3 DCS Variants

Two variants of the DCS scheme, based on the CSLT architecture: DCS-ICSLT and DCS-ACSLT, are explored in this section.

DCS-ICSLT

DCS-Independent Choke Sensor Lookup Table (ICSLT) refers to the architecture where each error tag occupies an independent tuple. There is no correlation between the tuples in



Cycle	Cycle-1 [0]	Cycle-1 [1]	....	Cycle-1 [n-2]	Cycle-1 [n-1]
$(\text{opc-owm})_e$	$(\text{opc-owm})_p$	$(\text{opc-owm})_p$		$(\text{opc-owm})_p$	$(\text{opc-owm})_p$
$(\text{opc-owm})_e$	$(\text{opc-owm})_p$	$(\text{opc-owm})_p$		$(\text{opc-owm})_p$	$(\text{opc-owm})_p$
⋮	⋮	⋮		⋮	⋮

Fig. 3.6: An Associative Choke Sensor Lookup Table (ACSLT) with associativity value n. *opc* refers to opcodes. The subscripts e and p refer to the errant and previous cycle, respectively.

an ICSLT. Same opcode-OWM pair can occupy multiple tuples, if the previous cycle opcode-OWM pairs are mutually exclusive. Such an architecture resembles a fully associative cache. In Section 3.5.2, the prediction accuracy of different entry-sized ICSLTs is discussed.

One drawback of DCS-ICSLT scheme is the redundancy in storing error tags. A considerable number of tuples in the CSLT are observed to be occupied by redundant errant opcode-OWM pairs. This redundancy restricts the optimal utilization of the CSLT space. To address this limitation, a second scheme is proposed, detailed next.

DCS-ACSLT

DCS-Associative Choke Sensor Lookup Table (ACSLT) refers to the table architecture where each tuple houses all the recorded error instances for a single errant instruction. The table resembles a set associative cache, where the errant opcode-OWM forms the set and the previous cycle opcode-OWM pairs form the lines of the set. This architecture eliminates the redundant storage of recurring errant opcode-OWM pairs, which in turn saves space. The *associativity* of an ACSLT determines the number of error instances each tuple can store. Figure 3.6 shows an n-associative ACSLT. In Section 3.5.2, the prediction accuracy of ACSLTs is discussed with different associativity values.

3.3.4 DCS Stages

DCS mechanism operates in three interlinked stages, that are discussed next.

Choke Sensing

This is practically the learning phase of DCS. For a given chip, the error causing instruction sequence is expected to sensitize the same choke path on every occurrence. However, the errant instruction sequence might be different for different chips of the same design due to the randomness in PV distribution. Hence, by allowing each unique timing error instance to occur atleast once, the system learns the choke paths of the specific chip for the given application.

When a timing error is sensed, it is recorded in the CSLT. The table entries are managed dynamically, in the form of a Random Access Memory (RAM). For the sensing mechanism, double-sampling flip-flops are used at the output of potential sensitized non-critical paths, similar to Razor [19]. The potential paths are identified by the method suggested by Lak et al. [33]. A buffer is implemented to hold the opcode-OWM values from decode (De) stage till the writeback (WB) stage, in order to preserve the previous cycle data until the current opcode completes execute (Ex) stage.

When the CSLT becomes full, pseudo-LRU (Least Recently Used) technique is used to evict existing tags and make space for new entries. Pseudo-LRU harvests the benefit of LRU while avoiding its complex hardware design. After a power off state, when the system boots, the table is populated during the warm-up period, thereby eliminating the need for any memory augmentation. The CSLT does not increase the critical path, since the lookup is performed in parallel to the normal operation of the pipestages between De and Ex stages. Further, the logic depths of the paths in CSLT are too low to create prospective critical paths.

Choke Error Recovery

After a timing violation has been recorded, the Choke Controller initiates a pipeline flush to erase the current cycle results of all the pipe stages. The flush is followed by an instruction replay to repopulate the stages and resume the normal flow of execution. This recovery mechanism incurs a penalty of P cycles, where P is the number of pipe stages.

Timing Error Avoidance

This is the adaptive phase, where stall cycles are used to avoid imminent errors predicted by CSLT. For each cycle, the corresponding opcode-OWM is looked up in the CSLT during the decode stage, to avoid repeated timing errors. An error causing opcode-OWM sequence is likely to repeat its behavior in subsequent cycles, under same operating conditions. Even a small change in the operating condition is expected to repeat some of the previous timing errors, while creating some new instances. These new instances will be duly recorded in the CSLT for future references. The Bloom Filter [34] mechanism is implemented to lookup the table using the tag described in Section 3.3.2. If a match is found, a timing error is expected to occur again. Hence, the Choke Controller stalls the progress of subsequent instruction before the execution stage, for an additional cycle, and guarantees the propagation of correct results thereafter.

3.3.5 Error Handling

Depending on the type of error encountered during the table lookup, the penalty cycle count varies. A false positive match returned by the CSLT results in an additional stall cycle penalty. However, a false negative match incurs higher penalty cycles, as the Choke Controller initiates a pipeline flush and instruction replay when the error is finally encountered in the execution stage.

3.4 Methodology

In this section, the rigorous cross-layer methodology that is employed to establish the potency of the proposed technique in a 11-stage pipeline is discussed. Figure 3.7 depicts the multiple layers that are broadly outlined in the following sections. Section 3.4.1 discusses the estimation technique for the delay distribution of FinFETs as well as the methodology to estimate process variation. In Section 3.4.2, the steps taken to gather inputs from real world benchmarks for the in-house STA tool are described. Section 3.4.3 discusses the circuit level implementation of the proposed technique.

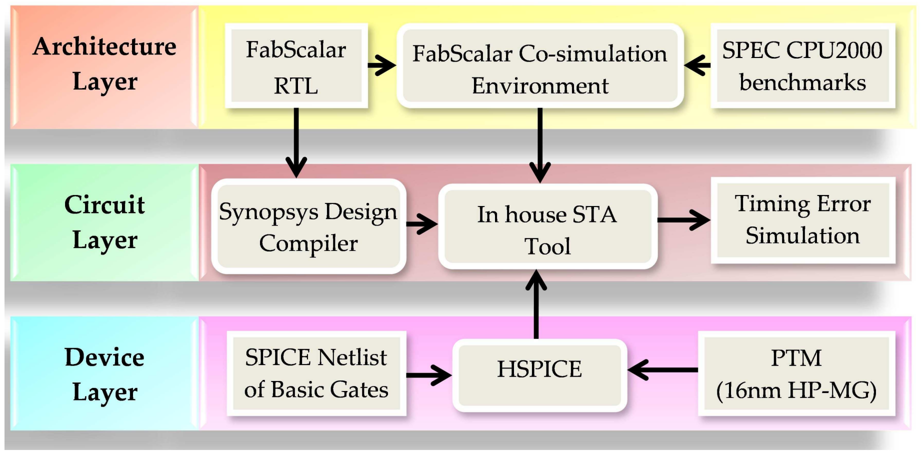


Fig. 3.7: Cross-layer simulation and analysis flow.

3.4.1 Device Layer

In order to estimate the gate delay distribution of FinFETs at different operating voltages, HSPICE models of fundamental logic gates based on the Predictive Technology Model (PTM) are simulated for 16nm high-performance multigate devices [35]. VARIUS [14] and VARIUS-NTV [5] models are chosen for PV at STC and NTC, respectively. To model the impacts of PV in FinFETs, the analysis presented in [36] is taken into account. For the simulations, oxide thickness is varied by 20%, fin thickness is varied by $\pm 10\%$ and channel length is varied by $\pm 12\%$. Monte Carlo simulation is performed for 10000 instances, to evaluate the means and standard deviations of propagation delay distributions of the basic gates at STC and NTC regimes. These propagation delay values are then used in the circuit layer simulation, as discussed in Section 3.4.3.

3.4.2 Architecture Layer

FabScalar infrastructure [37] is used to perform the architectural simulation. The Core-1 configuration is chosen for the experiments. It has a 11-stage out-of-order superscalar pipeline that is capable of fetching, issuing and committing 4 instructions in each cycle. In the execution stage, a choke sensing mechanism and a tactic to insert stall cycles are employed. For all the other stages, a pipe stage flush and instruction replay procedure similar to Razor [19] are applied. Six SPEC CPU2000 benchmarks are used, that typify real world applications. The cycle-by-cycle input values for all these benchmarks are obtained that are used for the statistical timing analysis discussed in Section 3.4.3.

3.4.3 Circuit Layer

The circuit layer simulation has three stages. In the first stage, the ALU RTL are synthesized using Synopsys Design Compiler (SDC). For this study, the focus is singularly on the choking in the execute stage of pipeline. 15nm NanGate Open Cell Library for FinFETs [31] is used to perform the synthesis. The clock frequency is set at 250MHz and the design is optimized with respect to power. The components of the DCS schemes are designed and synthesized, to estimate the energy consumption at NTC, for energy efficiency analysis later (discussed in Section 3.5.5). In the second stage, the synthesized netlist and the input values for all the benchmarks are fed into the in-house STA Tool. The process variation induced gate delay values obtained from HSPICE simulation are also incorporated into the STA Tool. By the end of this stage, the propagation delay values of the sensitized paths in each cycle are procured for all the benchmarks. In the third stage, these delay values are utilized to perform the timing error simulation for diverse schemes. The runtime and number of penalty cycles encountered by each benchmark for each scheme are calculated to present a comparison of efficacy among them. Finally, the area and wirelength overheads are evaluated using Cadence Encounter tool [38].

3.5 Experimental Results

In this section, the experimental results are illustrated in comparison to popular existing techniques like Razor and HFG. Section 3.5.1 highlights the comparative schemes, while Section 3.5.2 discusses the prediction accuracies of the two variants of the proposed scheme for multiple configurations. Sections 3.5.3, 3.5.4 and 3.5.5 present a comparative study of the recovery penalties, performance metric and energy-efficiency gains of the techniques, respectively, with the schemes listed in Section 3.5.1. Section 3.5.6 discusses the area and power overheads of this technique.

3.5.1 Comparative Schemes

- **Razor**: This timing speculation based error detection and recovery scheme [19], detects a timing error by a double-sampling flip-flop at the end of each pipeline stage. The recovery mechanism flushes the pipeline stages and initiates an instruction replay. However, unlike the proposed scheme, Razor cannot predict errors. Hence, it sustains repeated recovery penalty.

- **HFG**: This scheme proactively prevents timing errors [21]. It adaptively modifies the guardband, to account for PVTA (Process, Voltage, Temperature, Aging) variations throughout the device lifetime. Sampled data from the sensors, are used to train the model for guardband prediction. But, in order to avoid even worst case errors, the applied guardband increases the overall execution time. Therefore, though there is no recovery penalty, the performance and power overheads are considerably high.

- **DCS-ICSLT**: The first proposed scheme senses timing errors caused by choke points. It then uses this knowledge to avoid similar potential timing errors in future. Moreover, it also reduces the timing penalty caused by repeated pipeline flush and instruction replay. In this scheme, each tuple consists of only one error tag.

- **DCS-ACSLT**: The second proposed scheme is same as above, except, the CSLT structure. In this scheme, each tuple in the table consists of a single errant instruction opcode-OWM pair and multiple previous-cycle opcode-OWM pairs.

3.5.2 Prediction Accuracy

Figure 3.8 shows the prediction accuracy of DCS for different entry-sizes of the ICSLT. Each of the benchmarks are simulated for 1 million cycles and, the errors and prediction counts for ICSLTs with different sizes of entries are recorded. The varied prediction accuracies displayed by different benchmarks is owing to the variance in number of unique error instances among them. The figure clearly shows that prediction accuracy varies minimally

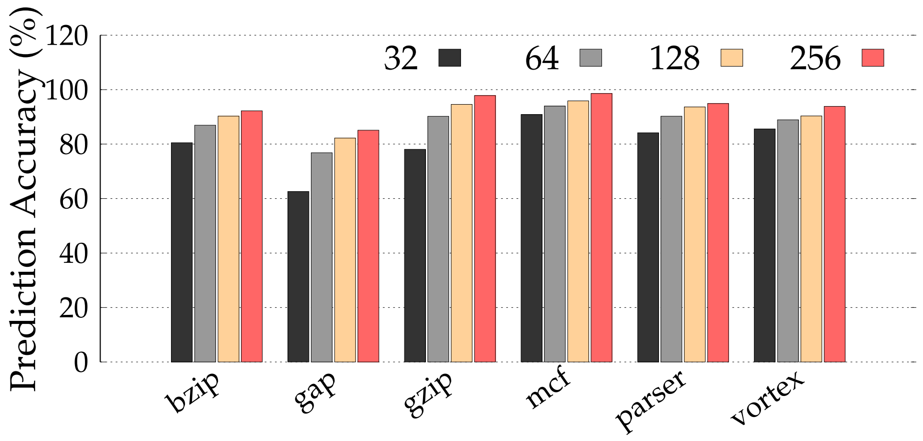


Fig. 3.8: Prediction accuracy of DCS across all benchmarks for 32, 64, 128 and 256 entries in CSLT.

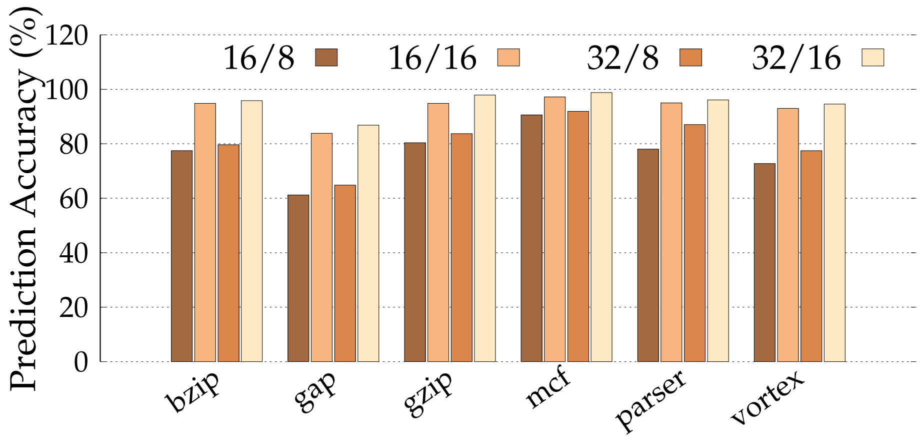


Fig. 3.9: Prediction accuracy comparison of 4 combinations of ACSLT:16/8 - entries-16 associativity-8, 16/16 - entries-16 associativity-16, 32/8 - entries-32 associativity-8, 32/16 - entries-32 associativity-16

from 128 to 256 entries for most of the benchmarks. So, for a fair trade-off between prediction accuracy and space efficiency, the ICSLT size of 128-entries is considered for further evaluations.

Similarly, Figure 3.9 shows the prediction accuracy of DCS for different size combinations of ACSLT. Observing the total number of errant cases in all the benchmarks under consideration, the results for only four combinations are chosen to be presented here. Smaller combinations exhibit worse results for prediction accuracy and larger combinations

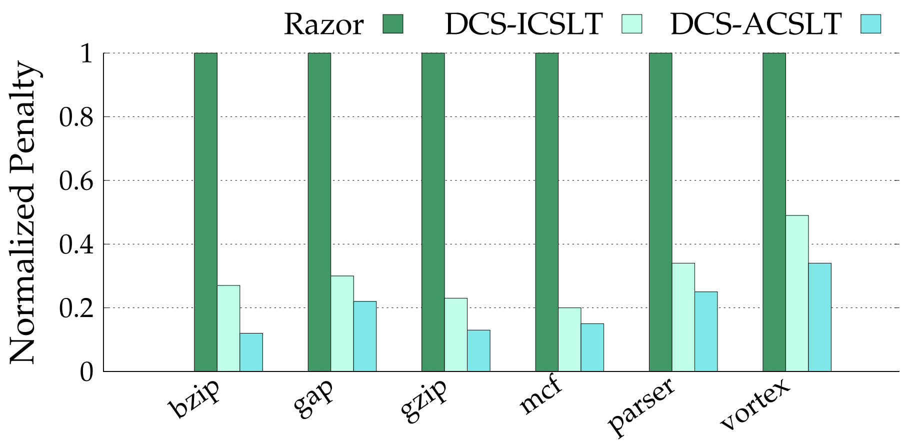


Fig. 3.10: Recovery penalty comparison (normalized to Razor values) of Razor and DCS schemes for different applications (lower is better).

tend to increase the hardware overhead extensively. In ACSLT, the variation in prediction accuracy is dictated not only by the number of unique error instances, but also the associativity value. It is clear from the results that 32-entries 16-way structure gives the maximum prediction accuracy. This combination is used for further comparisons with other schemes.

3.5.3 Recovery Penalty Comparison

Figure 3.10 presents a comparison between the recovery penalties incurred by Razor, DCS-ICSLT and DCS-ACSLT schemes, normalized to Razor values. Notably, HFG has been left out of the comparison, since it does not allow timing errors to occur by providing focused timing guardbands, and hence, no penalty is incurred. For all the benchmarks, DCS-ICSLT and DCS-ACSLT show substantially reduced penalties. Early choke sensing allows to prevent the timing error and the corresponding instruction replay with the insertion of the stall cycle. Applications like *mcf* and *gzip* show about 80% reduction in penalty, while *vortex* shows only about 50% reduction for DCS-ICSLT. DCS-ACSLT, on the other hand shows about 90% reduction in penalty for *gzip* and about 65% reduction in *vortex*. This phenomenon is attributed to the imbalance in the unique errant instruction count in the applications. While *mcf* has a small set of recurring error causing instructions, *vortex* has a large set of unique error causing instructions. However, *mcf* does not exhibit

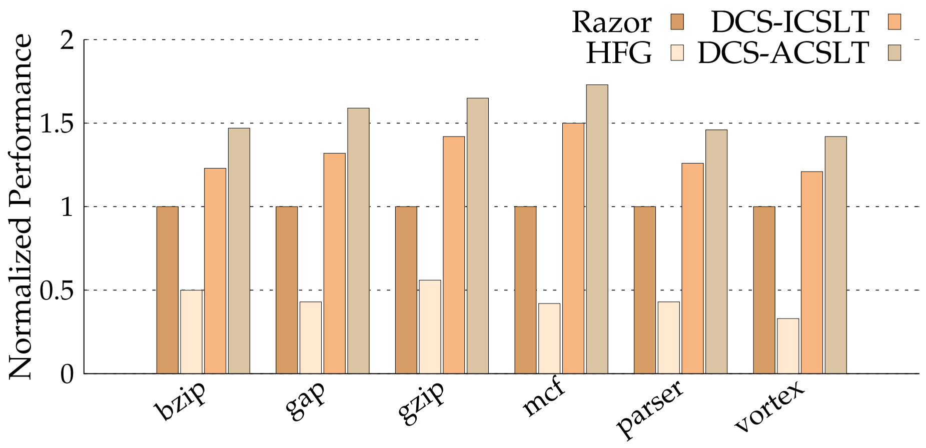


Fig. 3.11: Performance comparison of three comparative schemes for different applications (higher is better).

substantial reduction from DCS-ICSLT to DCS-ACSLT, owing to the smaller set of unique error instances, which are largely sensed and avoided by DCS-ICSLT. Similar effects are observed in the evaluation of performance and energy efficiency of the applications in the following sections.

3.5.4 Performance Gain

Figure 3.11 depicts the performance gains achieved by the proposed techniques as compared to Razor and HFG. The performances of all the schemes are normalized with respect to Razor. HFG is seen to have the worst performance, among the three schemes, for all the applications. The reason being, HFG simply increases the clock period based on the guardband range. So even for a few instances of potential timing error, the overall runtime is increased in HFG. Razor, however, performs better as it allows the timing errors to occur and then initiates the recovery mechanism to avoid faulty data propagation. DCS-ACSLT shows the best performance among all the schemes, for all the applications. Both the DCS schemes use the knowledge of previous timing error instances to foretell potential instances in subsequent cycles. Apart from the opcode itself, DCS schemes also consider the operand width for choke point sensing, thereby harvesting fine-grained knowledge of the errant opcode. But the additional performance gain in DCS-ACSLT stems from the

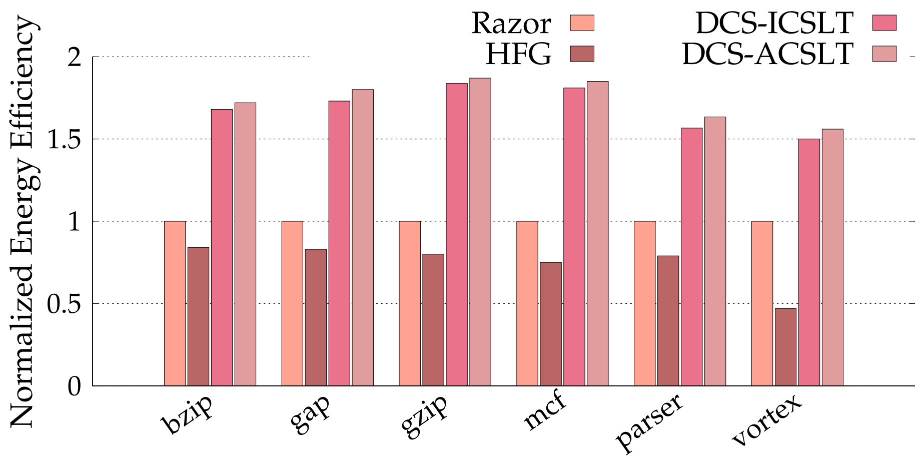


Fig. 3.12: Comparison of energy efficiency of the comparative schemes for different benchmarks (higher the better).

reduced execution time. The reduction is owing to the improvised lookup table structure and, thereby, the expedited lookup mechanism.

Among the benchmarks, mcf–with the minimum number of unique tuples–displays 50% performance improvement for DCS-ICSLT and 73% for DCS-ACSLT, compared to Razor. On an average, DCS-ICSLT shows 30% performance improvement as compared to Razor and 150% as compared to HFG. Whereas, DCS-ACSLT offers 55% and about 200% improvement, on an average, as compared to Razor and HFG, respectively.

3.5.5 Energy Efficiency Gain

Figure 3.12 displays the energy efficiency improvement achieved by DCS, for all the applications, as compared to the other schemes. The energy efficiency values are measured as the inverse of energy-delay products (EDP). EDP for each benchmark is calculated as $P_{\mathrm{ave}} \times t_{\mathrm{exec}}$, where P_{ave} is the average power consumption of the system and t_{exec} is the total execution time of each benchmark. All the energy efficiency values are normalized with respect to that of Razor. Among all the applications, *gzip* is observed to be most energy efficient with about 83% improvement in DCS-ICSLT and 87% in DCS-ACSLT over Razor. The anomaly in the relative performance and energy efficiency values of *gzip* and *mcf* is owing to the fact that total error count of *gzip* is lesser than *mcf*, but the number of unique error instances is more. The greater number of tuples reduce the scope of performance gain from DCS schemes as compared to *mcf*. However, the overall execution time being lesser than *mcf*, the energy efficiency improvement is slightly higher. Notably, the energy efficiency gain from DCS-ACSLT is not as significantly higher than DCS-ICSLT as in case of performance gain. This phenomenon is a consequence of the slight increase in hardware overhead due to a 32-entries ACSLT with associativity value 16. On an average, DCS-ICSLT exhibits about 60% improvement in energy efficiency over Razor and 90% improvement over HFG across all the benchmarks. On the contrary, DCS-ACSLT presents about 73% improvement in energy efficiency over Razor and about 103% improvement over HFG.

3.5.6 Overheads

In terms of gate counts, DCS-ICSLT uses 1553 additional gates, while for DCS-ACSLT, 3241 gates are used. The CSLT is composed of 567 and 2255 gates for DCS-ICSLT and DCSACSLT, respectively. The remaining gate counts are used by controller, buffer and the lookup logic for CSLT.

The area and power overheads incurred by DCS schemes are also negligibly small. While the area and wire-length overheads of DCS-ICSLT are 0.23% and 0.77% of that of the entire processor pipeline, respectively, the power overhead is 0.85% of the core power. Though the overheads for DCS-ACSLT are higher than DCS-ICSLT, yet they are trivial with respect to the overall design. Area, wirelength and power overheads for DCS-ACSLT are 0.48%, 0.85% and 1.2%, respectively. The power overhead results have been included in the energy efficiency results in Figure 3.12.

CHAPTER 4

TRIDENT: COMPREHENSIVE CHOKE ERROR MITIGATION IN NTC SYSTEMS

4.1 Background and Contributions of This Work

The emergence of the power constrained Internet of Things (IoT) applications has prompted the research community to focus on the development of low-power devices. Consequently, Near Threshold Computing (NTC)—where the supply voltage is marginally higher than the threshold voltage—has emerged as a promising design paradigm. But the overwhelming performance degradation ($\sim$10$\times$) and reliability concerns (due to $\sim$20$\times$ gate delay variation), at NTC, undermine the energy efficiency gains from the reduced supply voltage [39].

One such significant reliability concern is a *Choke Point* [8]. In this work, some critical design challenges posed by choke points at NTC, and the inefficacy of conventional techniques in tackling them, are demonstrated.

A choke point is a small set of process variation (PV) affected gates (or a single gate) that practically dominates the delay of the entire path in which it occurs. Notably, choke points are discernible only in the sensitized paths of a fabricated chip, and are capable of substantially deviating the path delay in either direction. Recent works have uncovered the potency of choke points in causing critical path delay violations [40]. However, the potency of choke points in causing minimum timing violations has remained unexplored. In this work, the significance of minimum timing violations caused by choke points in NTC systems is underlined.

Minimum timing violations are avoided in most Super Threshold Computing (STC) systems by inserting buffers in short delay paths [19]. But, this work shows that enhanced PV sensitivity at NTC can transform buffers, like other logic gates, into potential choke points. These *choke buffers*, i.e., buffers acting as choke points, can cause minimum timing violations, due to significantly reduced gate delay. Since buffers constitute an important design criteria for many timing speculation based error mitigation techniques [19, 20, 41], choke buffers pose a consequential challenge to their efficiency at NTC. Therefore, to eliminate the risk of choke buffers, *Trident* — a novel comprehensive timing error mitigation technique against choke points at NTC, is proposed.

This is the first work that analyzes the potency of choke points in causing minimum timing violation, and thereby, reveals the drawbacks of adopting popular timing error mitigation techniques in tackling them,at NTC.

The precise contributions in this work are:

- The potency of choke points in causing minimum timing violations in a processor pipeline is analyzed in Section 4.2. Moreover, it is shown that the problem, though insignificant at STC, is extremely prominent at NTC.

- The choke error patterns are explored and the governing factors for choke errors are determined (Section 4.2).

- The limitations of buffer insertion technique, to tackle minimum timing violations, at NTC are highlighted (Section 4.2). Consequently, the inefficacy of adopting popular STC timing error mitigation techniques at NTC is established.

- *Trident*, a comprehensive timing error resilient technique against choke points, that eliminates the risk of choke buffers, is proposed (Section 4.3).

- Finally, it is demonstrated that the performance and energy efficiency gains, with the proposed technique, are significant at 1.3× and 1.1× over Razor [19], respectively.

4.2 Motivation

In this section, choke point induced minimum timing violations at NTC are investigated. In Section 4.2.1, the unique characteristics of choke points are briefly described. Next, in Section 4.2.2, the significance of choke point induced minimum timing violations are discussed. Subsequently, the experimental methodology and results for this motivational analysis are described in Sections 4.2.3 and 4.2.4, respectively. Finally, in Section 4.2.6, the challenges of a choke error resilient system design at NTC are presented, thereby underlining the limitations of adopting popular timing error mitigation techniques for the same.

4.2.1 Background

Choke points are byproducts of the fabrication process. Therefore, their occurrence and impacts vary chip to chip, even for the same design. *Choke errors* (i.e., timing violations/errors caused by choke points), being perceivable only when the corresponding paths are sensitized, are greatly dependent on the input vectors to the system [40, 41]. Common PV modelling techniques are not sufficient to evaluate these impacts. For example, Monte Carlo simulation effectively determines the static delay variation of logic gates, but fails to incorporate the contributions of input vectors in sensitizing these gates. As a result, the divergence of path delay variation across the system, with respect to diverse applications, remains obfuscated in these models. Thus, a dynamic PV modelling technique is necessary for analyzing choke points.

Critical path delay violations by choke points have been recently addressed [40, 41]. In the next section, the focus is on the minimum timing violations caused by choke points, and their significance in designing a comprehensive and efficient choke error mitigation technique is highlighted.

4.2.2 Facets of Choke Points Induced Minimum Timing Violations

Figure 4.1 illustrates a choke point induced minimum timing violation. A minimum timing violation occurs when the *minimum path delay constraint* [1] is breached. PV can affect the gate delay both positively and negatively [7]. Substantial reduction in gate delay can diminish the overall delay of the path containing the corresponding gate. In Figure 4.1,

[1] *Minimum path delay constraint* is the lower bound of the path delay, to avoid data corruption.

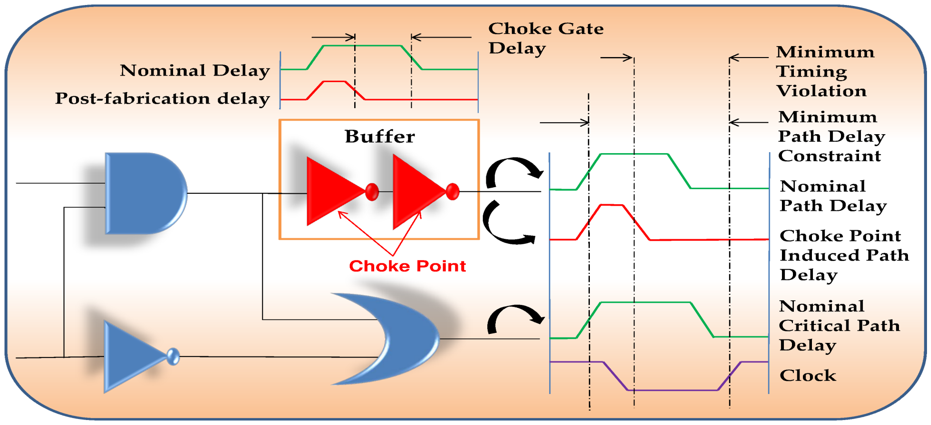


Fig. 4.1: A minimum timing violation caused by Choke Point induced delay reduction in a buffered short path

due to the choke buffer, the corresponding path delay is reduced beyond the minimum path delay constraint.

Besides latching erroneous value at the output node, minimum timing violations can also compromise the detection of maximum timing violations. For example, double sampling based error mitigation techniques [19, 20, 40, 41] rely on buffers to avoid data corruption in short delay paths. The concept of choke buffers, renders these techniques inefficient at NTC. To elucidate the relation between choke buffers and minimum timing violations, PV-induced path delay variations are experimentally analyzed, at both STC and NTC. The details of the experimental setup are described next.

4.2.3 Methodology

To explore the role of choke points in causing minimum and maximum timing violations at NTC and STC, an instruction level analysis is performed on a RISC-based processor pipeline. The study is focussed on the execute (EX) stage, as it is observed to be deeply affected by aggressive voltage and frequency scaling [19]. Further, a larger variation of sensitized paths in the EX stage is observed, compared to other pipestages. A set of 15 arithmetic and logic instructions are simulated, with a wide range of operands such as to replicate real world applications. The EX stage is a part of the Core-1 configuration of the FabScalar infrastructure [37]. The EX stage is augmented with the buffers, and

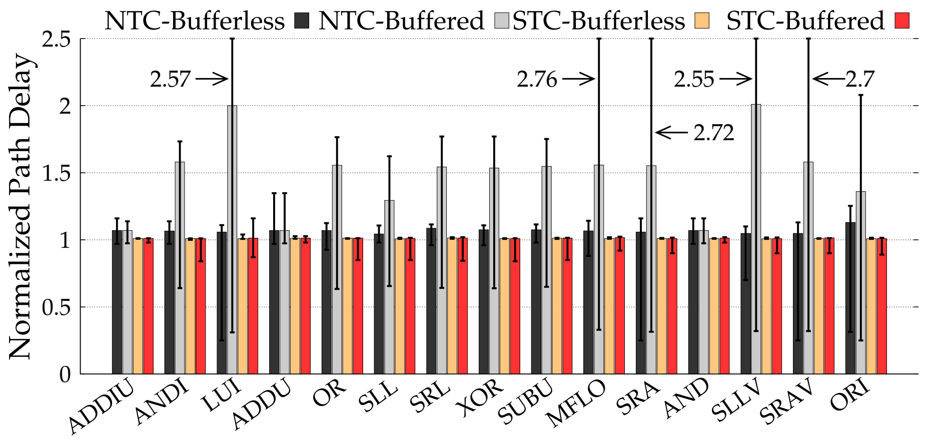


Fig. 4.2: path delay variations at STC and NTC for a given set of instructions. The minimum delay paths are simulated with and without buffers to study the effect of PV on buffered paths. The error bars denote the minimum path delay and maximum path delays. The values are normalized with respect to corresponding PV-free path delays.

is synthesized using Synopsys Design Compiler (SDC) and the FinFET OpenCell library from NanGate [42]. The number of buffers is calculated as described in [19]. The basic logic gates are simulated in HSPICE using the 16nm multigate models from Predictive Technology Models (PTM) [35]. To model the effects of PV on FinFETs, the analysis presented in [43] is used. Finally, a statistical dynamic timing analysis is performed on the synthesized EX stage, using the in-house tool, to study the choke point induced timing violations per cycle.

4.2.4 Results

Figure 4.2 illustrates a comprehensive picture of the path delay variations caused by choke points in buffered and bufferless delay paths at NTC and STC operating conditions. In the buffered version of the EX stage, the short delay paths are augmented with buffers to satisfy the minimum path delay constraint. The maximum, minimum and average path delay variations are normalized with respect to their respective PV-free path delay variants. For all the instructions in Figure 4.2, the variations at NTC are remarkably greater than their STC counterparts. The number of gates acting as choke points is limited to 2% of the total gate count to demonstrate that such limited presence can cause a visible impact.

But the crux of this analysis is that, almost all the instructions show greater variations in the EX stage with buffered delay paths, at NTC. Large minimum path delay variations are observed in 12 out of 15 instructions in Figure 4.2. Especially, instructions like MFLO and SLLV display over 60% reduction in minimum path delay in the buffered EX stage at NTC, as opposed to about 10% reduction in the bufferless counterpart. This observation clearly indicates that the manifestation of choke buffers essentially defeats the purpose of buffers in short delay paths, by causing minimum timing violations.

However, instructions like LUI and SRA show a greater minimum path delay variation in bufferless EX stage at NTC. This anomaly can be attributed to the limited buffer requirement of the short delay paths sensitized by these instructions. As a result, though the absence of buffers cause minimum timing violations for these instructions, the buffers do not transform into choke buffers even with delay reduction. Contrary to the observations at NTC, buffered and bufferless EX stages at STC do not show a significant difference in path delay variations. This phenomenon asserts that choke buffers have restricted impact at STC. The observations, while corroborating the effectiveness of buffer insertion technique at STC, underlines the inefficacy of the same at NTC. The dramatic variations in the path delays, at NTC, lay the foundation for diverse timing error patterns. In the next section, the timing error patterns of the instructions are explored and the factors governing the choke errors at NTC are deduced.

4.2.5 Patterns & Factors of Choke Error

Figure 4.3 shows the normalized occurrence patterns for 8 different instructions. As the figure portrays, while many of the instances of each instruction cause either maximum or minimum timing violation, a considerable share of occurrences do not cause any timing error. Therefore, an instruction, that caused a timing error once, cannot be blindly predicted to cause error in every single occurrence. The maximum and minimum timing errors caused by the instructions are the direct impact of the path delay variations discussed in Section 4.2.4. Instructions like MFLO and LUI show that about 70% of their occurrences cause maximum timing errors; whereas, for instructions like OR and ADDU, about 55%

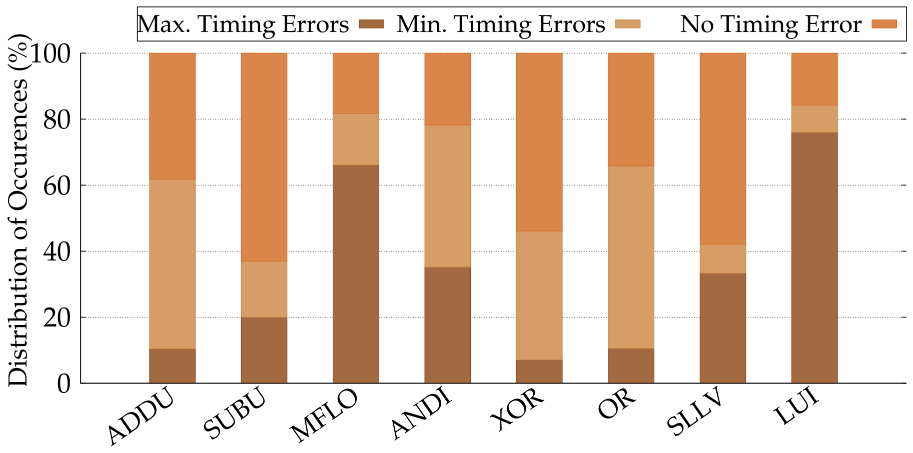


Fig. 4.3: Distribution of erroneous and error-free occurrences of diffrent instructions.

occurrences cause minimum timing errors. On the other hand, instructions like ANDI and SUBU show almost equal share of maximum and minimum timing errors. Remarkably, SUBU also shows more than 50% error-free occurrences. These observations bring us to the inference, that the choke errors, caused by minimum or maximum timing violations, cannot be characterised by only a single instruction opcode. A deeper analysis reveals that the choke errors are dictated by a sequence of two consecutive instruction opcodes. The sequence comprise the initializing instruction and the sensitizing instruction. The initializing instruction determines the output state of the path in the cycle immediately preceding the errant cycle. The sensitizing instruction tends to change the output state, thereby triggering the choke error. Further analysis of the choke error patterns reveal that the sizes of the instruction operands play a pivotal role, as discussed next.

Figure 4.4 illustrates the distribution of the timing errors with respect to the operand sizes of the instructions. To determine the size of the operands, simply the position of the leftmost set bit is identified. For example, in a 32-bit operand, if the leftmost set bit lies in the two higher bytes, the size is considered "Large" (denoted by 1); otherwise, the size is determined to be "Small" (denoted by 0). As the figure depicts, both operand size variants are responsible for maximum and minimum timing violations. However, the figure clearly shows that for both maximum and minimum timing errors, "Large" operands have a greater

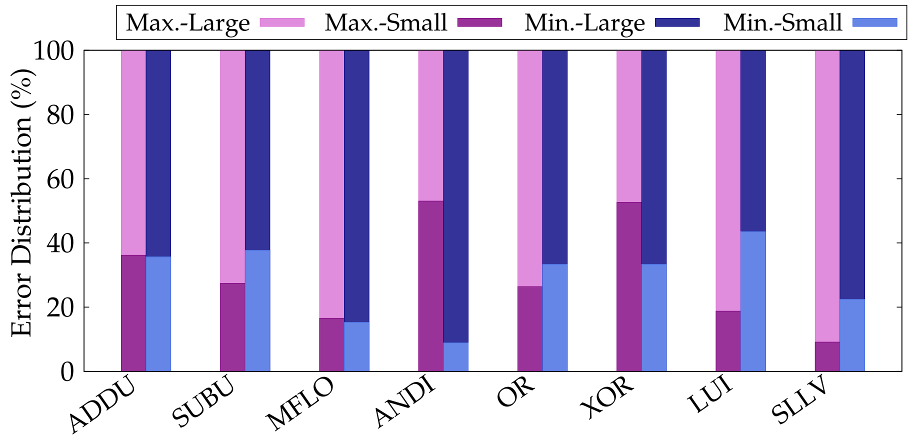


Fig. 4.4: A comparative study of maximum and minimum timing errors with respect to operand sizes of errant instructions.

influence. The larger the operand, the more number of paths are sensitized in the circuit, thereby increasing the probability of activating more choke points. Across all benchmarks, "Large" operands constitute about 70% of the minimum timing errors. Notably, 91.09% minimum timing errors caused by instruction ANDI are contributed by "Large" operands. On the contrary, instructions like LUI and XOR display about equal share of either operand sizes in causing minimum timing errors and maximum timing errors, respectively. This characteristic can be attributed to the operand patterns of these two instructions. Even the "Small" operands have many set bits (i.e,"1"), which in turn sensitize a large number of circuit paths. Hence, the number of maximum timing errors caused by the "Small" operands for these two instructions are high.

In the light of these divergent observations, the design challenges for a comprehensive choke error resilient system at NTC are deduced and discussed next.

4.2.6 Challenges with Choke Points

The observations in Section 4.2.4 reveal three main challenges. Firstly, the overall path delay variations in choke point affected systems are higher at NTC than STC. Conclusively, it can be said that the effects of choke points at NTC are more severe, than they are at STC. Secondly, effects of choke points are not restricted to causing maximum timing violations.

The considerable variations in minimum path delays at NTC constitute a fair share of choke errors. Finally, addition of buffers in short delay paths does not ensure minimum timing violation aversion, at NTC. This failure of buffer insertion technique reduces the scope of several error mitigation methodologies [19, 20] at NTC.

A key research question here is *how to design a comprehensive timing error mitigation technique that is capable of addressing all the above challenges posed by choke points at NTC?* To address this question, *Trident*, a novel comprehensive error mitigation technique for tackling the varied impacts of choke points at NTC, is proposed. The proposed scheme is detailed in the next section.

4.3 TRIDENT: A comprehensive choke point resilient technique

This section elaborates the design and functionality of the proposed technique, *Trident*. Sections 4.3.1 and 4.3.2 present the objective and overview of Trident, respectively. Section 4.3.3 highlights a key insight behind the Trident model. In Section 4.3.4, the error tags are analyzed; and finally, in Sections 4.3.5, 4.3.6 and 4.3.7 the components and mechanisms of Trident are discussed, respectively.

4.3.1 Objective of Trident

Trident aims at tackling all timing errors caused by choke points. Unlike the Razor based detection technique proposed in [40, 41], Trident considers that all logic gates, including the gates forming the buffers, are potential choke points. Consequently, this technique eliminates the use of buffer insertion technique to avoid minimum timing violations. Instead, it uses a detection mechanism for illegal transitions (discussed in Section 4.3.5) to account for all timing violations caused by choke points. On the basis of the number of illegal transitions in one clock cycle, timing errors caused by choke points can be broadly categorised into two classes:

- **Single Error (SE):** These are isolated timing violations (minimum or maximum), which are neither preceded nor followed by any other timing error event, within a

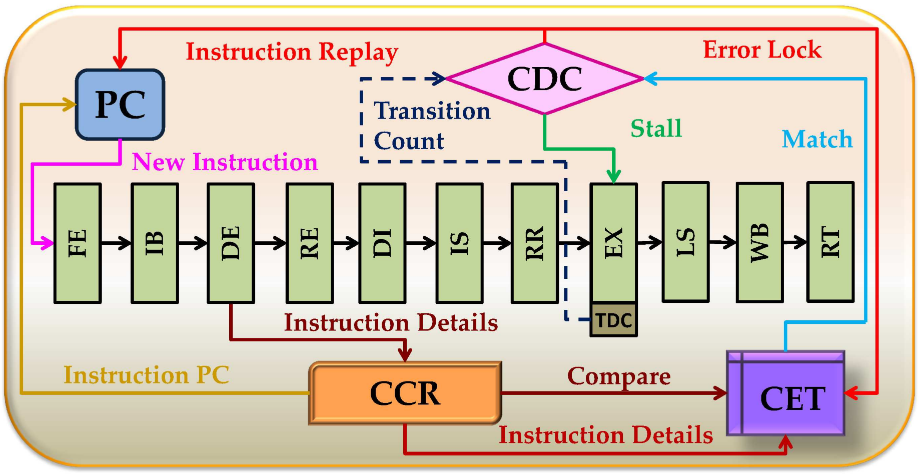


Fig. 4.5: Design blocks and data flow of Trident. The EX (Execution) pipestage is under scrutiny.

given clock cycle. They are characterised by a single illegal transition of the data signal, in one clock cycle.

- **Consecutive Error (CE):** These errors are caused by back-to-back timing violations in a single clock cycle. These errors are characterised by two illegal transitions of the data signal, in one clock cycle. A CE comprising a minimum timing violation followed by a maximum timing violation is not possible, because the corresponding illegal transitions would span over more than one clock cycle (discussed in detail in Section 4.3.6). Therefore, a CE is one in which a maximum timing violation is immediately followed by a minimum timing violation.

In the next section, a brief overview of the Trident design components and flow is presented.

4.3.2 Design Overview

Figure 4.5 illustrates the block diagram of Trident, featuring the flow of operations. To learn the individual choke point signature of a chip, Trident allows the first occurrence of an error. The *choke error detection mechanism* is orchestrated by the *Transition Detector and Counter (TDC)* and the *Choke Detection Controller (CDC)*. First, an error is detected by the TDC and classified by the CDC. The error instance is then recorded in the *Choke Error Table (CET)*, using an *Error ID (EID)*, for future references. Next, in order to launch the *choke error correction mechanism*, the CDC initiates a pipeline flush and indicates the program counter (PC) to perform an instruction replay. The errant instruction address is provided by the *Choke Clearance Register (CCR)*, which holds the details of the instruction in the pipeline between decode (DE) and writeback (WB) stages. The *choke error avoidance mechanism* is a specialty of Trident. This mechanism enables the system to avoid recurrent detection and correction of repeated errors. Earlier researches have shown that errant instructions tend to repeat their behavior [23, 24] (discussed in Section 4.3.3). Trident exploits this intriguing circuit-architectural property. For the avoidance mechanism, the newest instruction in the CCR is compared to the entries in the CET, for potential matches. If a match is found, the CET informs the CDC of the pipestage and class of the error. The CDC, in turn, decides on the number of stall cycles to be inserted into the pipestage, based on the class of the error. The stall cycle halts the progress of the subsequent instructions in the pipeline, while allowing the specified pipestage an additional cycle to complete error-free execution. The choke error avoidance mechanism being a key feature of Trident, in the next section, the architectural insight behind effective choke error prediction is discussed.

4.3.3 Effective Choke Error Prediction Principle

Earlier researches have shown that dynamic instances of a static instruction tends to sensitize the same path [24, 44]. Instruction sequence locality, as well as, operand value locality, for a given instruction, are some of the vital factors contributing to the stated correlation. Researchers have cited many well known instruction execution characteristics which demonstrate this correlation [45]. For example, a small set of load instructions has been observed to sensitize the same path during each execution instance and cause repeated cache failures [46, 47]. With this insight, the Trident model is armed with timing error prediction mechanism to avoid repeated errors. The key tool for this prediction and avoidance mechanism is the EID. The features of EID are described in detail, next.

4.3.4 Error ID (EID)

The EID is a combination of the factors governing an error. The EID comprises the initializing and sensitizing vector [23], the operand sizes of the vectors [40, 41], the class of error (described in Section 4.3.1) and the errant pipestage. The class, on the other hand, is determined on the basis of the number of illegal transitions caused by the timing violations, as discussed in Section 4.3.5. The size of the CET is denoted by the maximum number of EIDs it can hold. In the following sections, the interactions among the hardware components of the Trident are elaborately described to elucidate the role of EIDs in choke error detection and avoidance mechanisms.

4.3.5 Components of Trident

There are four hardware components that regulate the three mechanisms of *Trident*. The functionality of these components in each of the different mechanisms are elaborated next.

Choke Error Table (CET)

The CET is used to record the error instances, encountered during the choke error detection mechanism, in the form of EIDs. The table is structured in the form of a Random Access Memory (RAM). During the choke error avoidance mechanism, the details (discussed in Section 4.3.4) corresponding to the latest instruction in the CCR are compared against the EIDs. The Bloom filter [34] is used for parallelized lookup in the CET. Consequently, timing overhead due to EID lookup is eliminated. In case there is a match, the CET intimates the class and pipestage of the error to the CDC, for appropriate measures. If the CET fills up and a new entry is to be made, Pseudo-LRU (Least Recently Used) policy is followed for replacement.

Transition Detector and Counter (TDC)

TDCs work only during the choke error detection mechanism. Every pipestage, between decode (DE) and writeback (WB), is provided with a TDC. Each TDC comprises a double-edged flip-flop [48], to detect both rising and falling transitions. The TDC is controlled by the *detection clock*, similar to the one described in [49]. The *detection clock* deactivates the TDC only for a small interval around the rising edge of the system clock. During the active phase, the TDC detects and counts the illegal transitions of the output data. When deactivated, the TDC feeds the count to the CDC, for classification. Any transition during this small interval is not flagged as illegal.

Choke Clearance Register (CCR)

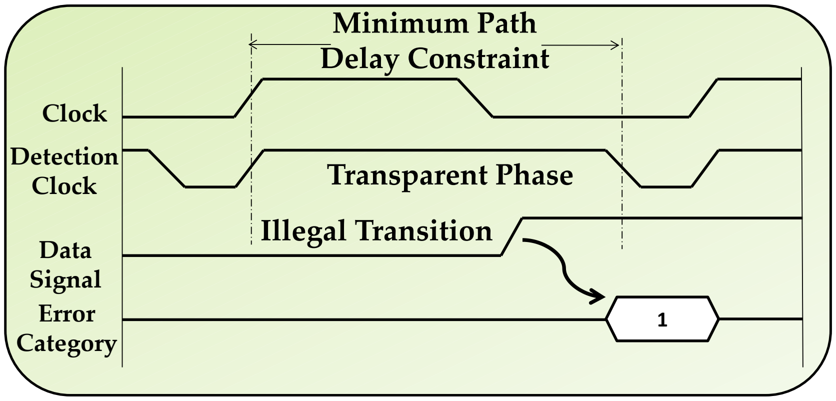
This is a form of instruction buffer that stores the opcode, operand sizes and PC value of each instruction between DE and WB stage. During the detection mechanism, it provides the instruction details for the EID. In the *choke error avoidance mechanism* (discussed in Section 4.3.7), it provides the details for comparison to the EID. Further, in the *choke error correction mechanism* (discussed in Section 4.3.6), it provides the PC with the errant instruction address for instruction replay.

Choke Detection Controller (CDC)

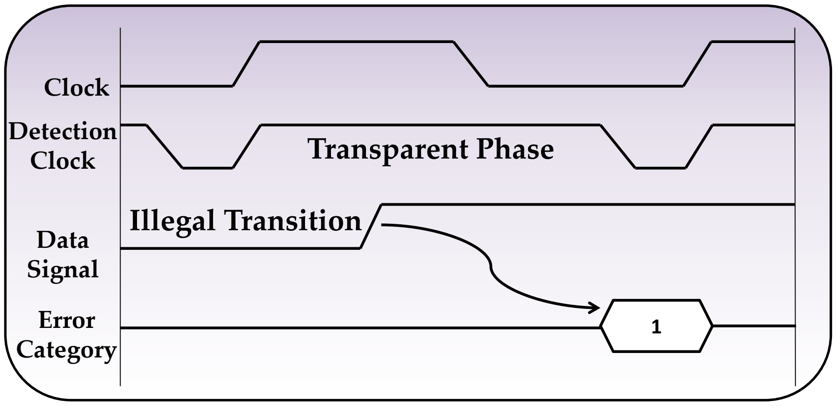
This component spearheads the entire design flow of Trident. For the *choke error detection mechanism* (discussed in Section 4.3.6), the CDC classifies the errors on the basis of the TDC count. It then logs the error instances in the CET. The CDC is also responsible for the *choke error correction mechanism* (discussed in Section 4.3.6), where it performs a pipeline flush and indicates the PC to perform an instruction replay. For the *choke error avoidance mechanism*, the CDC inserts the necessary number of stall cycles (as discussed in Section 4.3.7) based on the error class information provided by the CET.

4.3.6 Choke Error Detection & Correction Mechanisms

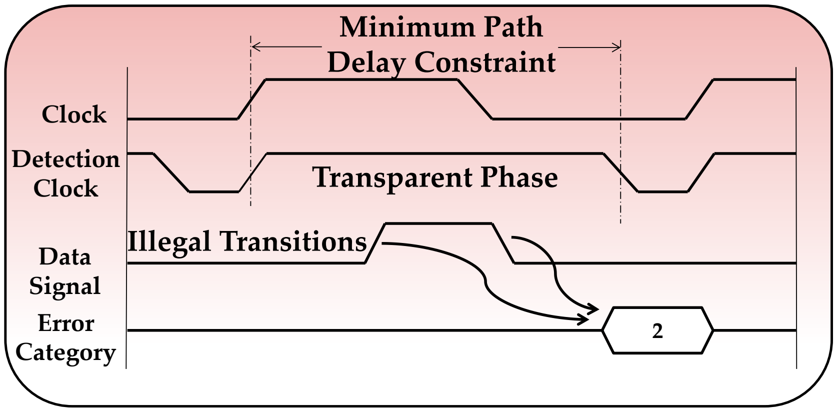
Figure 4.6 portrays the choke error detection mechanism for the three different types of choke errors. Figures 4.6a and 4.6b show the two varieties of SE, caused by minimum and maximum timing violations, respectively. For both the cases, there is only a single illegal transition during the transparent phase of one *detection clock* cycle. The TDC



(a) Min. Error



(b) Max. Error



(c) Max.-Min. Error

Fig. 4.6: The figures show the signal transitions during the three different types of errors. The transitions during the transparent phase of the detection clock are flagged as illegal. The double-edge triggered flip flop increases the counter in the TDC for each illegal transition in one clock cycle. The low pulse in the detection clock resets the counter for the next cycle.

counts these illegal transitions and with the rising edge of the *detection clock*, the count value is latched and fed to the CDC. This count indicates the class of the detected error. Contrarily, in Figure 4.6c, which illustrates the CE, there are two illegal transitions during the transparent phase of one *detection clock* cycle. The first transition is the ramification of a maximum timing violation; while the second one marks a subsequent minimum timing violation. However, a minimum timing violation followed by a maximum timing violation does not constitute a CE. In such a case, the illegal transitions do not occur within a single cycle of the detection clock. Therefore, the minimum timing violation is detected and classified as an SE before the next violation occurs.

Nevertheless, the correction mechanism is same for both classes of errors. Once an error is detected and classified, the CDC flushes the entire pipeline to remove the corrupt data. Next, the PC retrieves the errant instruction address from the CCR and launches an instruction replay. The detection and correction mechanism incurs as many penalty cycles as the number of pipestages. During replay, the *choke error avoidance mechanism* (discussed next) prevents rerun of the detection and correction mechanism, thereby saving the recurrent penalty cycles.

4.3.7 Choke Error Avoidance Mechanism

The choke error avoidance mechanism relies on timely insertion of stall cycles into the processor pipeline. The count of stall cycles to be inserted is dictated by the error class. Avoiding SE requires a single stall cycle. As shown in Figure 4.6a, the early transition due to the minimum timing violation, corrupts the previous instruction results. At this instant, the CCR provides the previous cycle instruction details to the CET to record the error. For the avoidance mechanism, a stall cycle is inserted after this instruction. The stall cycle ensures that the results from the previous instruction is latched successfully before it gets changed by the minimum timing violation. At this point the error is avoided as the results from both the instructions are correctly latched at the end of the pipestage. On the other hand, an SE caused by maximum timing violation is avoided by allowing an extra cycle of execution time, for the current instruction, in the form of a stall cycle. Contrary to an

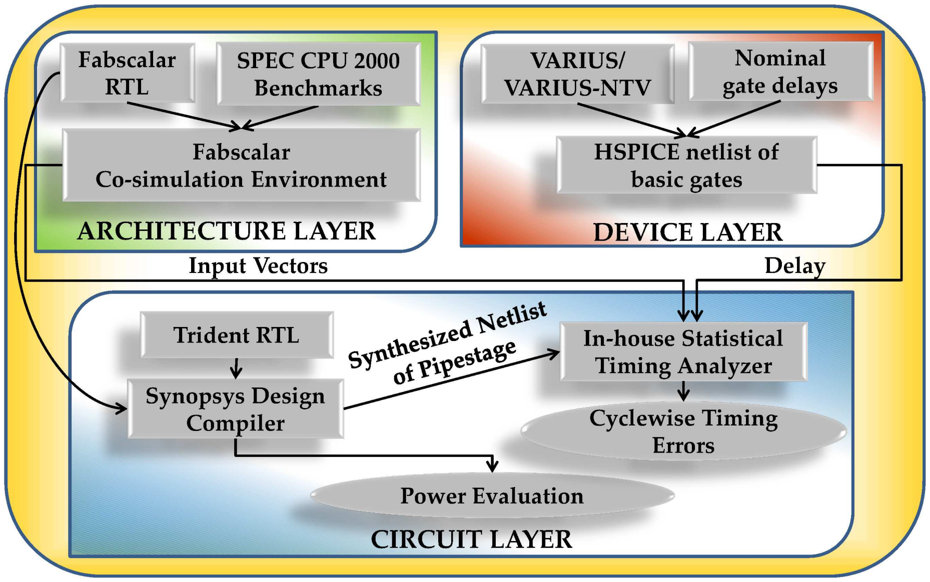


Fig. 4.7: Interaction among the layers in the cross-layer methodology.

SE, a CE causes a chain of data corruptions, shown in Figure 4.6c. Consequently, two stall cycles are required to mitigate a CE. The first cycle mitigates the maximum timing violation by allowing additional clock period; while the second cycle avoids the data corruption due to minimum timing violation, by holding on to the data for one extra cycle. Therefore, the illegal transition count recorded during detection mechanism also signifies the required number of stall cycles for the avoidance mechanism. This avoidance mechanism is followed for each predicted error, as well as, the false positive matches. However, the false negative matches are handled by the detection and correction mechanisms.

In the next section, the multi-layer methodology for implementation and assessment of Trident is described.

4.4 Methodology

Figure 4.7 portrays the cross-layer design methodology used in this work. In this section, each layer is described in detail.

4.4.1 Device Layer

In this layer, the focus is on determining the effects of voltage scaling and PV on the basic logic gate delays. The VARIUS [14] and VARIUS-NTV [5] models are used to estimate the effects of PV on the delays of basic logic gates at STC and NTC, respectively. To incorporate the effects of PV on FinFETs, the model presented in [36] is utilized. The delay values obtained from HSPICE simulations (discussed in Section 4.2.3) are used for timing analysis of the circuit, described in Section 4.4.3.

4.4.2 Architecture Layer

In this layer of design, six SPEC CPU2000 benchmarks [50] are simulated for 1 million cycles, using the FabScalar infrastructure [37], to generate the input vectors for the synthesized EX stage described in Section 4.2.3. Further, the EX stage RTL is augmented with the *Trident* design components described in Section 4.3.5. The augmented RTL and the input vectors are essential for the circuit synthesis and dynamic timing analysis in the circuit layer of design methodology (Section 4.4.3).

4.4.3 Circuit Layer

In this layer of design flow, the circuit synthesis and the timing analysis are performed. First, the augmented EX stage is synthesized using Synopsys Design Compiler [51] and the NanGate library as described in Section 4.2.3. Next, a statistical timing analysis is conducted with the in-house tool. The tool accepts the synthesized netlist, the input vectors and the logic gate delay values as inputs and generates a cyclewise sensitized path delay report. The effects of PV are incorporated in the logic gate delay values to emulate the effects of choke points. The deterministic gate delay values of the PV-free logic gates are provided by a pre-specified delay library. However, for the PV affected gates, a Gaussian delay distribution is provided for each type of gate and the tool selects random sample delay values from the given distribution each time. Combined with a random selection of PV affected gates for each run, this feature of the tool closely imitates the random choke delay signature of different chips. Finally, the path delay report from the tool is used to analyze the timing violations. Cadence SoC Encounter [38] is used to place and route the design, and thereby the overall area, wiring and power overheads are calculated.

4.5 Experimental Results

In this section, the efficacy of *Trident* is evaluated. The proposed technique is compared to two timing error detection and mitigation techniques, described in Section 4.5.1. Section 4.5.2 displays the distribution of choke error classes across the benchmarks as detected by *Trident*. Section 4.5.3 presents comparison of prediction accuracy for different configurations of the CET. In Sections 4.5.4, 4.5.5 and 4.5.6, the incurred penalties, performance improvements and energy efficiency gains of *Trident* are assessed, respectively, in comparison to the schemes discussed in Section 4.5.1. Finally, in Section 4.5.7, the overheads associated with *Trident* are presented.

4.5.1 Comparative Schemes

- **Razor:** This technique detects maximum timing errors in combinational paths with the use of a shadow latch [19]. Razor employs buffer insertion to avoid minimum timing violations in short delay paths, and has no error prediction mechanism. This scheme is the baseline.

- **Online Clock Skew Tuning (OCST):** This technique combines timing speculation with clock skew tuning [20]. Clock skews are adjusted dynamically, according to the timing error occurrences at runtime. This technique also relies on buffers to avoid minimum timing errors.

- **Trident:** This technique adapts to the choke point signature of a specific chip and dynamically tackles both minimum and maximum timing violations. Most importantly, it is equipped with choke error avoidance mechanism, which greatly impacts its performance and energy efficiency.

4.5.2 Error Distribution

Figure 4.8 shows the distribution of SEs and CEs across different benchmarks. Buffers are inserted in the short delay paths (as described in [19]) to analyze the effects of choke

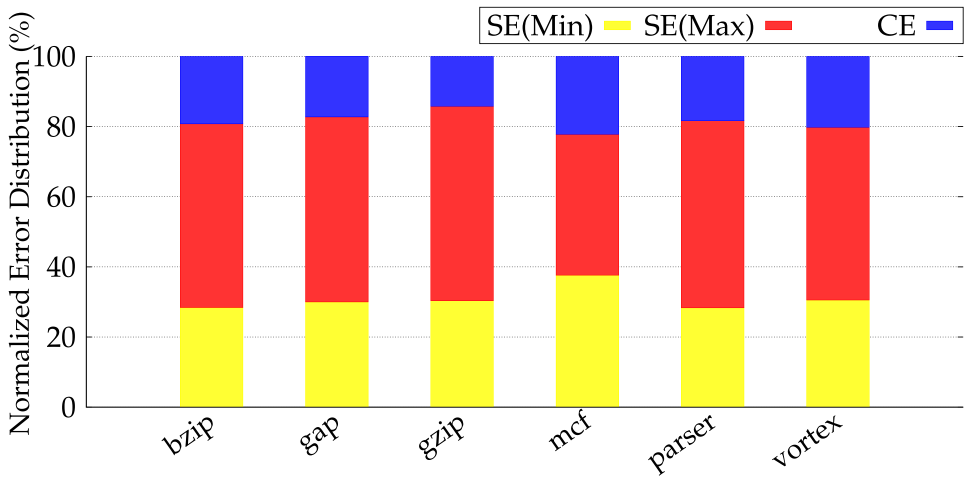


Fig. 4.8: Distribution of SE and CE for each benchmark. SE are caused by either minimum timing violations [SE(Min)] or maximum timing violations [SE(Max)].

buffers. In order to account for all the errors, the choke error avoidance mechanism is disabled during this experiment. As the figure shows, about 80% of all the errors are SEs. For a deeper analysis, the SEs are distinguished into minimum and maximum timing violations. It is observed that about 37.5% of the SEs are constituted of minimum timing violations. Further, considering the CEs, minimum timing violations clearly make up a significant fraction of the choke errors detected by Trident. In the following section, the impact of various CET sizes on the prediction accuracy of these diverse choke errors is studied.

4.5.3 Table Size vs. Prediction Accuracy

The size of the CET, i.e., the number of EID entries in a CET, is the key factor in determining the choke error prediction accuracy of Trident. The prediction accuracy, in turn, determines the penalty cycle count, performance and energy efficiency of Trident, as discussed in the next few sections. Figure 4.9 shows the choke error prediction accuracy for different benchmarks with respect to different CET sizes. It is observed that, for all the benchmarks, there is noticeable rise in prediction accuracy from an average of 81.75% at 32-entries size to 92.88% at 128-entries size. However, the increment in accuracy from 128-

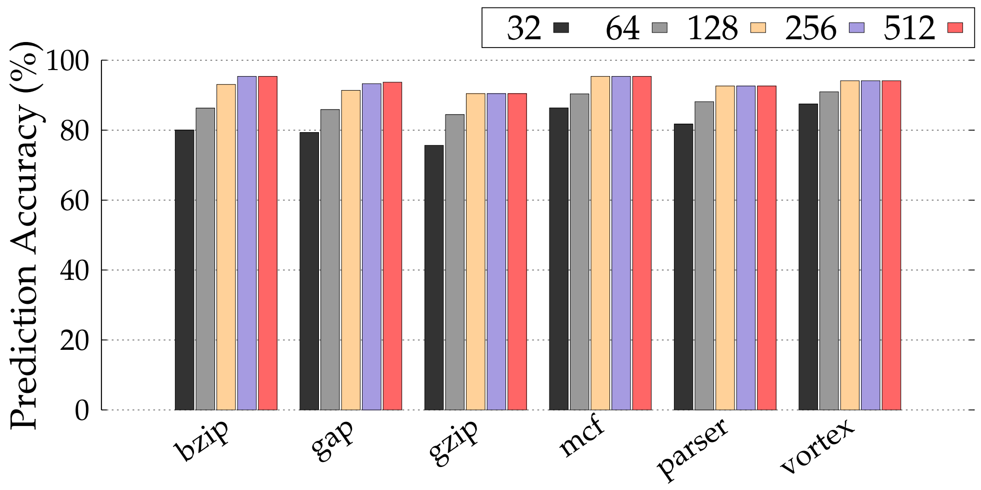


Fig. 4.9: The choke error prediction accuracy comparison for different entry sizes of the Choke Error Table (CET).

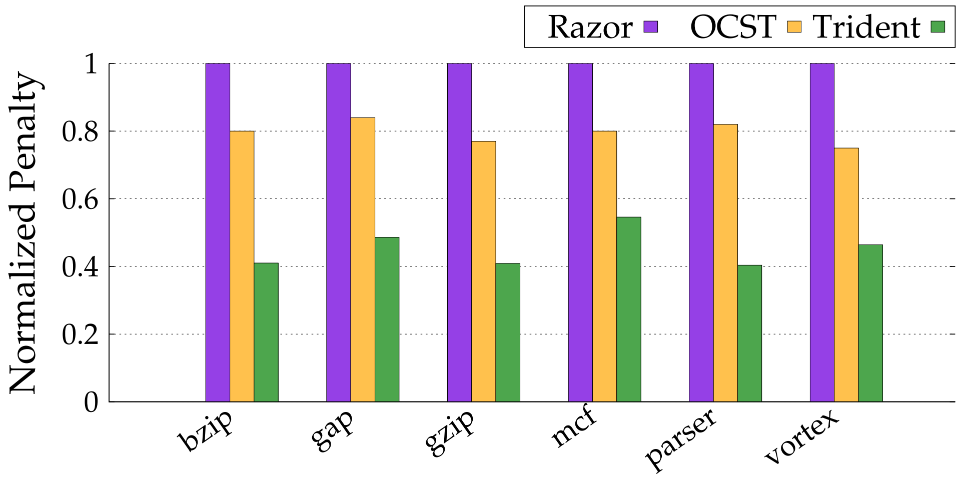


Fig. 4.10: Normalized penalty cycle count comparison of the schemes, for each benchmark.

entries to 512-entries is barely 2.3%, across all benchmarks. Therefore, 128-entries CET is chosen for further experimental evaluations of Trident. Next, the penalties incurred across the three comparative schemes are discussed.

4.5.4 Penalty Cycle Comparison

Figure 4.10 shows the normalized penalty cycles incurred by each benchmark for the three comparative schemes. In all the cases, Trident performs the best with least number

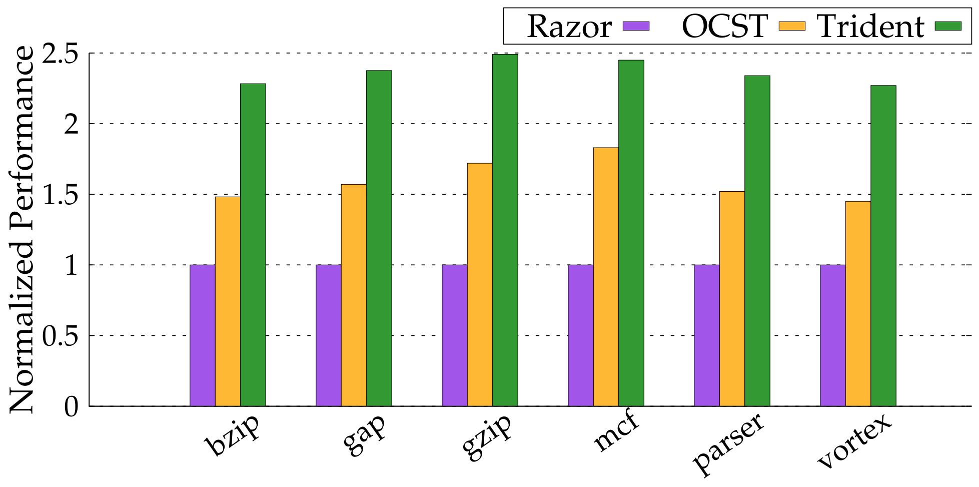


Fig. 4.11: Performance impact comparison of Trident with Razor and OCST. (Higher is better.)

of penalty cycles. The reduction in penalty cycles is achieved due to the error avoidance mechanism of Trident (discussed in Section 4.3). The error avoidance mechanism limits the penalty cycles incurred due to repeated error corrections. OCST performs better than Razor owing to the fact that the clock skew is tuned after every 100,000 cycles, considering the timing errors encountered in that tuning interval [20]. On an average, OCST incurs 20% less penalty than Razor, while Trident incurs 60% and 43.75% less penalty compared to Razor and OCST, respectively. Notably, the penalty cycle count for Trident considers both minimum and maximum timing errors, while those for Razor and OCST consider only maximum timing errors. In the subsequent sections, the impacts of these penalty cycle counts on the performance and energy efficiency of each scheme are studied.

4.5.5 Performance Comparison

Figure 4.11 illustrates the performance impact of each of the comparative schemes. The performance is evaluated on the basis of the penalty cycles incurred in detecting and recovering from errors and the resultant impact on execution time of each application. All the performance values are normalized with respect to Razor values. OCST offers about 57.7% improvement in performance over Razor. However, *Trident* offers about 1.37× and

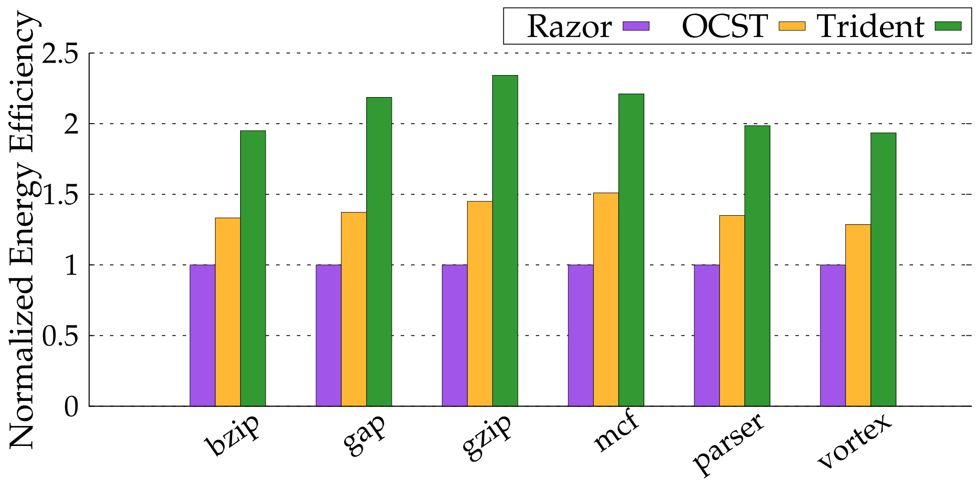


Fig. 4.12: Energy efficiency comparison of Trident with Razor and OCST. (Higher is better.)

0.49× improvement over Razor and OCST, respectively. This substantial performance gain in *Trident* can be attributed to its ability to detect both minimum and maximum timing violations and to avoid repeated error occurrences. The latter considerably reduces the recovery penalty cycles (discussed in Section 4.5.4) and consequently, the execution time of the application. An intriguing phenomenon is observed regarding *gzip* and *mcf*. Both of these benchmarks display high performance gains, but for different reasons. *mcf* harbors the benefit of error avoidance, owing to the small number of unique error instances across all three categories. Contrarily, *gzip* has more unique error instances. But, the total number of errors is lesser in *gzip*, compared to *mcf*, and it has the smallest share of CEs. Therefore, *gzip* benefits from the reduced number of stall cycles due to CEs and the overall reduction in penalty cycles.

4.5.6 Energy Efficiency Comparison

Figure 4.12 shows the energy-efficiency gain achieved by *Trident* over Razor and OCST. The energy efficiency is evaluated as the reciprocal of energy-delay product (EDP). All the values are normalized with respect to Razor values. OCST offers an average gain of 38.35% in energy efficiency over Razor. *Trident* displays an additional 51.85% improvement, on an average, over OCST. This massive energy efficiency gain is contributed by the reduced recovery penalty (discussed in Section 4.5.4), as well as, the reduced overheads (as discussed in Section 4.5.7). Compared to all the benchmarks, *gzip* shows the maximum gain of 0.54× over OCST and 1.34× over Razor.

4.5.7 Hardware Overheads

The overheads are calculated after the placement and routing of the EX stage, augmented with the Trident components. The area, power and wirelength overheads of Trident, with respect to the unaltered EX stage, are 9.48%, 12.76% and 11.21%, respectively. Compared to the entire pipeline, the area, power and wirelength overheads are 0.97%, 1.58% and 1.12%, respectively.

The minimal overhead values also advocate for the scalability of the design. On one hand, a nominal increase in the size of the CET can considerably improve the prediction accuracy, and therefore the power-performance, in presence of greater number of unique timing errors. On the other hand, a larger circuit with more paths in every pipestage would translate into additional area overhead due to more double-edged latches, only, for the newer paths. An increase in the number of pipestages would, however, require additional TDCs. Notably, in all the above cases, the overheads due to CCR or the CDC, the components common across the design, remain constant. As a result, the relative overheads for larger circuits will be lesser than those mentioned above.

CHAPTER 5
CONCLUSION

This dissertation addresses a pivotal reliability challenge in NTC systems — *choke points*. Choke points are a post-fabrication outcome that are random in their occurrence, as well as, impacts. They are the small set of PV affected gates which can drastically alter the delay of the paths in which they occur. This alteration can be either positive or negative, i.e., the path delay can dramatically increase or decrease. Consequently, choke points are potential sources of both maximum and minimum timing violations.

This work demonstrates that conventional timing error detection and correction techniques are inefficient to tackle the unique challenges posed by choke points. The errors induced by choke points cannot be anticipated at design time. The choke errors are manifested when the paths containing the choke points are sensitized. Therefore, though a choke point once formed remains in the circuit forever, they are manifested occasionally. This dissertation addresses these peculiar characteristics of choke points. The works presented here deeply investigate the choke errors to determine the error patterns and their correlation to instruction-operand value locality. The error patterns guide the design of the two dynamically adaptable proactive EDAC techniques proposed in this dissertation.

Being the first technique to address choke errors, *Dynamic Choke Sensing (DCS)* adaptively tackles maximum timing violations at runtime. In addition to detection and correction, this technique predicts imminent errors. The error sensing significantly reduces the penalty and, therefore, maintains the intrinsic energy efficiency of NTC systems. Two variants of DCS, illustrate a trade-off between area and power-performance metrics. The second technique, *Trident*, tackles all the categories of choke errors, including maximum, minimum and maximum-minimum timing violations. Trident demonstrates that buffer insertion technique, commonly used to avoid minimum timing violations in very short delay paths, is an added predicament in the presence of choke points. Consequently, choke buffers, i.e., buffers acting as choke points, are introduced. Unlike DCS, which depends on double-sampling flip-flops for error detection, Trident monitors signal transitions to detect choke errors. In addition to choke error resilience, DCS-ICSLT, DCS-ACSLT and Trident offer performance improvements of 30%, 55% and 1.37×, respectively, over Razor. In terms of energy efficiency, DCS-ICSLT, DCS-ACSLT and Trident are 60%, 73% and 54% better than Razor, respectively. The overhead cost for all the three techniques are negligibly low.

REFERENCES

[1] *Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits*, 2010.

[2] N. Pinckney, K. Sewell, R. Dreslinski, D. Fick, T. M. udge, D. Sylvester, and D. Blaauw, "Assessing the performance limits of parallelized near-threshold computing," in *DAC*, 2012, pp. 1143–1148.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] H. Kaul, M. Anders, S. Mathew, S. Hsu, A. Agarwal, F. Sheikh, R. Krishnamurthy, and S. Borkar, "A 1.45ghz 52-to-162gflops/w variable-precision floating-point fused multiply-add unit with certainty tracking in 32nm CMOS," in *ISSCC*, 2012, pp. 182–184.

[5] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, and J. Torrellas, "Varius-ntv: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages," in *DSN*, 2012, pp. 1–11.

[6] M. Seok, G. Chen, S. Hanson, M. Wieckowski, D. Blaaw, and D. Sylvester, "Cas-fest 2010: Mitigating variability in near-threshold computing," in *J. Emerg Selec. Topics Cir. Sys*, vol. 1, no. 1, 2011, pp. 42–49.

[7] U. R. Karpuzcu, N. S. Kim, and J. Torrellas, "Coping with parametric variation at near-threshold voltages," *IEEE Micro*, vol. 33, no. 4, pp. 6–14, 2013.

[8] V. De, "Fine-grain power management in manycore processor and system-on-chip (soc) designs," in *Proc. of ICCAD*, 2015, pp. 159–164.

[9] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (ntv) design—opportunities and challenges," in *Proc. of DAC*, June 2012, pp. 1149–1154.

[10] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, 2010.

[11] U. R. Karpuzcu, A. A. Sinkar, N. S. Kim, and J. Torrellas, "Energysmart: Toward energy-efficient manycores for near-threshold computing," in *HPCA*, 2013, pp. 542–553.

[12] D. Fick, R. G. Dreslinski, B. Giridhar, G. Kim, S. Seo, M. Fojtik, S. Satpathy, Y. Lee, D. Kim, N. Liu, M. Wieckowski, G. K. Chen, T. N. Mudge, D. Blaauw, and D. Sylvester, "Centip3de: A cluster-based NTC architecture with 64 ARM cortex-m3 cores in 3d stacked 130 nm CMOS," *J. of Solid-State Circ.*, vol. 48, no. 1, pp. 104–117, 2013.

[13] A. Y. Dogan, J. Constantin, M. Ruggiero, A. Burg, and D. Atienza, "Multi-core architecture design for ultra-low-power wearable health monitoring systems," in *Proc. of DATE*, 2012, pp. 988–993.

[14] S. 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 Tran. on Semicond. Manufac.*, vol. 21, pp. 3 –13, 2008.

[15] S. K. Khatamifard, M. Resch, N. S. Kim, and U. R. Karpuzcu, "Varius-tc: A modular architecture-level model of parametric variation for thin-channel switches," in *ICCD*, 2016, pp. 654–661.

[16] M. S. Golanbari, S. Kiamehr, M. Ebrahimi, and M. B. Tahoori, "Variation-aware near threshold circuit synthesis," in *Proc. of DATE*, 2016, pp. 1237–1242.

[17] S. Kim and M. Seok, "Variation-tolerant, ultra-low-voltage microprocessor with a low-overhead, within-a-cycle in-situ timing-error detection and correction technique," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 6, pp. 1478–1490, 2015.

[18] M. S. Golanbari, S. Kiamehr, and M. B. Tahoori, "Hold-time violation analysis and fixing in near-threshold region," in *Power Timing, Model. Opt. Sim.*, 2016, pp. 50–55.

[19] D. Ernst, N. S. Kim, S. Das, S. Pant, R. R. Rao, T. Pham, C. H. Ziesler, D. Blaauw, T. M. Austin, K. Flautner, and T. N. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. of MICRO*, 2003, pp. 7–18.

[20] R. Ye, F. Yuan, and Q. Xu, "Online clock skew tuning for timing speculation," in *Proc. of ICCAD*, 2011, pp. 442–447.

[21] A. Rahimi, L. Benini, and R. K. Gupta, "Hierarchically focused guardbanding: an adaptive approach to mitigate PVT variations and aging," in *Proc. of DATE*, 2013, pp. 1695–1700.

[22] Z. Lak and N. Nicolici, "In-system and on-the-fly clock tuning mechanism to combat lifetime performance degradation," in *2011 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2011, San Jose, California, USA, November 7-10, 2011*, 2011, pp. 434–441.

[23] J. Xin and R. Joseph, "Identifying and predicting timing-critical instructions to boost timing speculation," in *Proc. of MICRO*, 2011, pp. 128–139.

[24] S. Roy and K. Chakraborty, "Predicting timing violations through instruction level path sensitization analysis," in *Proc. of DAC*, 2012, pp. 1074–1081.

[25] N. R. Pinckney, K. Sewell, R. G. Dreslinski, D. Fick, T. N. Mudge, D. Sylvester, and D. Blaauw, "Assessing the performance limits of parallelized near-threshold computing," in *Proc. of DAC*, 2012, pp. 1147–1152.

[26] *Ultralow-Power Design in Near-Threshold Region*, 2010.

[27] E. Krimer, R. Pawlowski, M. Erez, and P. Chiang, “Synctium: a near-threshold stream processor for energy-constrained parallel applications,” *Comp. Arch. Letters*, vol. 9, no. 1, pp. 21–24, 2010.

[28] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, “The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives,” in *Proc. of DAC*, 2014, pp. 185:1–185:6.

[29] A. Y. Dogan, J. Constantin, M. Ruggiero, A. Burg, and D. Atienza, “Multi-core architecture design for ultra-low-power wearable health monitoring systems,” in *Proc. of DATE*, 2012, pp. 988–993.

[30] J. Xin and R. Joseph, “Identifying and predicting timing-critical instructions to boost timing speculation,” in *Proc. of MICRO*, 2011, pp. 128–139.

[31] NanGate, http://www.nangate.com/?page_id=2328.

[32] K. Chakraborty, B. Cozzens, S. Roy, and D. M. Ancajas, “Efficiently tolerating timing violations in pipelined microprocessors,” in *Proc. of DAC*, 2013, pp. 1–8.

[33] Z. Lak and N. Nicolici, “In-system and on-the-fly clock tuning mechanism to combat lifetime performance degradation,” in *Proc. of ICCAD*, 2011, pp. 434–441.

[34] B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Comp. Arch. News*, vol. 13, pp. 422–426, 1970.

[35] S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao, “Exploring sub-20nm finfet design with predictive technology models,” in *Proc. of DAC*, 2012, pp. 283–288.

[36] H. R. Khan, D. Mamaluy, and D. Vasileska, “Simulation of the impact of process variation on the optimized 10-nm finfet,” *T. Electron Devices*, vol. 55, no. 8, pp. 2134–2141, 2008.

[37] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg, “FabScalar: composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template,” in *Proc. of ISCA*, 2011, pp. 11–22.

[38] S. Cadence, “Encounter user guide.”

[39] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. N. Mudge, “Near-threshold computing: Reclaiming moore’s law through energy efficient integrated circuits,” *Proc. of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[40] A. Bal, S. Saha, S. Roy, and K. Chakraborty, “Revamping timing error resilience to tackle choke points at ntc systems,” in *Proc. of DATE*, 2017, pp. 1020–1025.

[41] ——, “Dynamic choke sensing for timing error resilience in ntc systems,” *IEEE Trans. VLSI Syst.*, vol. 26, no. 1, pp. 1–10, 2018.

[42] M. Martins, J. M. Matos, R. P. Ribas, A. Reis, G. Schlinker, L. Rech, and J. Michelsen, “Open cell library in 15nm freepdk technology,” in *Proc. of ISPD*, 2015, pp. 171–178.

[43] B. C. Paul, S. Fujita, M. Okajima, T. H. Lee, H. Wong, Y. Nishi *et al.*, "Impact of a process variation on nanowire and nanotube device performance," *T. Electron Devices*, vol. 54, no. 9, pp. 2369–2376, 2007.

[44] H. Chen, S. Roy, and K. Chakraborty, "Darp: Dynamically adaptable resilient pipeline design in microprocessors," in *Proc. of DATE*, 2014, pp. 1–6.

[45] A. Sodani and G. S. Sohi, "Dynamic instruction reuse," in *Proc. of ISCA*, 1997, pp. 194–205.

[46] T. Sherwood, S. Sair, and B. Calder, "Predictor-directed stream buffers," in *Proc. of MICRO*, 2000, pp. 42–53.

[47] M. Annavaram, J. Patel, and E. Davidson, "Data prefetching by dependence graph precomputation," in *Proc. of ISCA*, 2001, pp. 52 –61.

[48] C.-C. Yu and K.-T. Chen, "A novel design of low-power double edge-triggered flip-flop," in *Proceedings of the 5th International Conference on Biomedical Engineering and Informatics*, 2012, pp. 1363–1366.

[49] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, "RazorII: In situ error detection and correction for PVT and SER tolerance," *J. of Solid-State Circ.*, vol. 44, no. 1, pp. 32–48, Jan. 2009.

[50] J. L. Henning, "Spec cpu2000: Measuring cpu performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28–35, 2000.

[51] D. Compiler, R. User, and M. Guide, "Synopsys," *Inc., see http://www. synopsys. com*, 2001.

CURRICULUM VITAE

Aatreyi Bal

Published Journal Articles

- Trident: Comprehensive Choke Error Mitigation in NTC Systems, Aatreyi Bal, Sanghamitra Roy and Koushik Chakraborty, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, issue 11, pp. 2195-2204, Nov 2018.

- Dynamic Choke Sensing for Timing Error Resilience in NTC Systems, Aatreyi Bal, Shamik Saha, Sanghamitra Roy and Koushik Chakraborty, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, issue 1, pp. 1-10, Jan 2018.

- Split Latency Allocator: Process Variation-Aware Register Access Latency Boost in a Near-Threshold Graphics Processing Unit, Asmita Pal, Aatreyi Bal, Koushik Chakraborty and Sanghamitra Roy, *Journal of Low Power Electronics*, vol. 13, pp. 419-427, 2017.

- TITAN: Uncovering the Paradigm Shift in Security Vulnerability at Near-Threshold Computing, Prabal Basu, Pramesh Pandey, Aatreyi Bal, Chidhambaranathan Rajamanikkam, Koushik Chakraborty and Sanghamitra Roy, *IEEE Transactions on Emerging Topics in Computing*, vol. 1, pp. 1-1, 2018.

- FIFA: Exploring a Focally Induced Fault Attack Strategy in Near-Threshold Computing, Prabal Basu, Chidhambaranathan Rajamanikkam, Aatreyi Bal, Pramesh Pandey, Trevor Carter, Koushik Chakraborty and Sanghamitra Roy, *IEEE Embedded Systems Letters*, vol. 10, issue 4, pp. 115-118, Dec 2018.

- SSAGA: SMs Synthesized for Asymmetric GPGPU Applications, Shamik Saha, Prabal Basu, Chidhambaranathan Rajamanikkam, Aatreyi Bal, Koushik Chakraborty and Sanghamitra Roy, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, pp. 49, 2017.

Published Conference Papers

- Trident: A comprehensive timing error resilient technique against choke points at NTC, Aatreyi Bal, Sanghamitra Roy and Koushik Chakraborty, in *Proc. IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018.
- Revamping timing error resilience to tackle choke points at NTC systems, Aatreyi Bal, Shamik Saha, Sanghamitra Roy and Koushik Chakraborty, in *Proc. IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.
- ACE-GPU: Tackling Choke Point Induced Performance Bottlenecks in a Near-Threshold Computing GPU, Tahmoures Shabanian, Aatreyi Bal, Prabal Basu, Koushik Chakraborty and Sanghamitra Roy, in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2018.