Trident: A Comprehensive Timing Error Resilient Technique against Choke Points at NTC

Aatreyi Bal, Sanghamitra Roy, Koushik Chakraborty

USU BRIDGE LAB, Electrical and Computer Engineering, Utah State University
bal.aatreyi@gmail.com, {sanghamitra.roy,koushik.chakraborty}@usu.edu

ABSTRACT

Near Threshold Computing (NTC) systems have been inherently plagued with heightened process variation (PV) sensitivity. *Choke points* are an intriguing manifestation of this PV sensitivity. In this paper, we explore the probability of minimum timing violations, caused by choke points, in an NTC system and, their non-trivial impacts on the system reliability. We show that conventional timing error mitigation techniques are inefficient in tackling choke point induced minimum timing violations. Consequently, we propose a comprehensive error mitigation technique, *Trident*, to tackle choke points, at NTC. *Trident* offers a 1.37× performance improvement and a 1.1× energy efficiency gain over Razor at NTC, with minimal overheads.

1. INTRODUCTION

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 (∼10×) and reliability concerns (due to ∼20× gate delay variation), at NTC, undermine the energy efficiency gains from the reduced supply voltage [1]. One such significant reliability concern is a *Choke Point* [7]. In this paper, we demonstrate some critical design challenges posed by choke points at NTC, and the inefficacy of conventional techniques in tackling them.

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. A recent work has uncovered the potency of choke points in causing critical path delay violations [2]. However, the potency of choke points in causing minimum timing violations has remain unexplored. In this paper, we underline the significance of minimum timing violations caused by choke points.

Minimum timing violations are avoided in most Super Threshold Computing (STC) systems by inserting buffers in short delay paths [8]. But, we show 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 [2, 8, 20], choke buffers pose a consequential challenge to their efficiency at NTC. Therefore, to eliminate the risk of choke buffers, we propose *Trident*, a novel comprehensive timing error mitigation technique against choke points at NTC.

To the best of our knowledge, *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.*

Our precise contributions in this paper are:

- We explore and analyze the potency of choke points in causing minimum timing violations in a processor pipeline (Section 2). Moreover, we show that the problem, though insignificant at STC, is extremely prominent at NTC.
- We highlight the limitations of buffer insertion technique, to tackle minimum timing violations, at NTC (Section 2). Consequently, we establish the inefficacy of adopting popular STC timing error mitigation techniques at NTC.
- Finally, we propose *Trident*, a comprehensive timing error resilient technique against choke points, that eliminates the risk of choke buffers (Section 3). The performance and energy efficiency gains, with our technique, are significant at 1.3× and 1.1× over Razor, respectively.

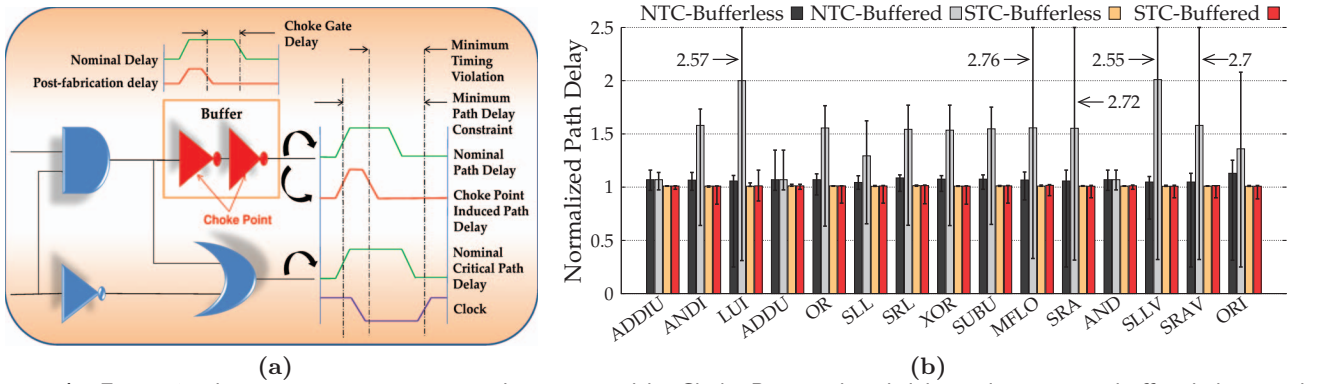
2. MOTIVATION

In this section, we investigate choke point induced minimum timing violations at NTC. In Sections 2.1, we briefly describe the unique characteristics of choke points. Next, in Section 2.2, we discuss the significance of choke point induced minimum timing violations. Subsequently, we describe our experimental methodology and results for this motivational analysis in Sections 2.3 and 2.4, respectively. Finally, in Section 2.5, we present the challenges of a choke error resilient system design at NTC, thereby underlining the limitations of adopting popular timing error mitigation techniques for the same.

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 [2]. 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.

978-3-9819263-0-9/DATE18/©2018 EDAA



(a) (b)

Figure 1: *Figure 1a shows a minimum timing violation caused by Choke Point induced delay reduction in a buffered short path. Figure 1b shows 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.*

Critical path delay violations by choke points have been recently addressed [2], with no consideration for potential minimum timing violations. In the next section, we focus on the minimum timing violations caused by choke points, and highlight their significance in designing a comprehensive and efficient choke error mitigation technique.

2.2 Facets of Choke Points Induced Minimum Timing Violations

Figure 1a 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 [11]. Substantial reduction in gate delay can diminish the overall delay of the path containing the corresponding gate. In Figure 1a, 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 [2, 8, 20] 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, we experimentally analyze PV-induced path delay variations, at both STC and NTC. Our experimental setup is described next.

2.3 Methodology

To explore the role of choke points in causing minimum and maximum timing violations at NTC and STC, we perform an instruction level analysis on a RISC-based processor pipeline. We focus our study on the execute (EX) stage, as it is observed to be deeply affected by aggressive voltage and frequency scaling [8]. Further, we observe a larger variation of sensitized paths in the EX stage, compared to other pipestages. We simulate a set of 15 arithmetic and logic instructions, 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 [4]. We augment the EX stage with the buffers, and synthesize it using Synopsys Design Compiler (SDC) and the FinFET OpenCell library from NanGate [13]. The number of buffers is calculated as described in [8]. We simulate the basic logic gates in HSPICE using the 16nm multigate models from Predictive Technology Models (PTM) [17]. To model the effects of PV on FinFETs, we use the analysis presented in [14]. Finally, we perform a statistical dynamic timing analysis of the synthesized EX stage, using our in-house tool, to study the choke point induced timing violations per cycle.

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

2.4 Results

Figure 1b 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. The maximum, minimum and average path delay variations are normalized with respect to the respective PV-free path delays. For all the instructions in Figure 1b, the variations at NTC are remarkably greater than their STC counterparts. We limit the gates acting as choke points to 2% of the total gate count to demonstrate the limited resources required to 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 1b. 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.

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. Contrary to our observations at NTC, buffered and bufferless EX stages at STC do not show a significant difference in path delay variations. Our observations, while corroborating the effectiveness of buffer insertion technique at STC, underlines the inefficacy of the same at NTC. In the light of these observations, we deduce the design challenges for buffer insertion technique in a choke error resilient system at NTC, discussed next.

2.5 Challenges with Choke Points

The observations in Section 2.4 reveals three main challenges. Firstly, the overall path delay variations in choke point affected systems are higher at NTC than STC. Con-

clusively, 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 delay at NTC constitute a fair share of choke point 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 [8, 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, we propose *Trident*, a novel comprehensive error mitigation technique for tackling the varied impacts of choke points at NTC. Our proposed scheme is detailed in the next section.

3. TRIDENT:A COMPREHENSIVE CHOKE POINT RESILIENT TECHNIQUE

In this section, we detail the design and functionality of our proposed technique, *Trident*. In Sections 3.1 and 3.2, we present the objective and overview of Trident, respectively. In Section 3.3, we analyze the error tags; and finally, in Section 3.4 we discuss the components of Trident.

3.1 Objective of Trident

Trident aims at tackling all timing errors caused by choke points. Unlike the Razor based detection technique proposed in [2], 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 3.4.2) 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. 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. A CE comprising a minimum timing violation followed by a maximum timing violation is not possible, because that would span over more than one clock cycle. The minimum timing violation will be detected before the next error instant. Therefore, a CE is one in which a maximum timing violation is immediately followed by a minimum timing violation. These errors are characterised by two illegal transitions of the data signal, in one clock cycle.

In the next section, we present a brief overview of the Trident design components and flow.

3.2 Design Overview

Figure 2 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. In the *choke error detection* stage, first an error is detected by the *Transition Detector and Counter (TDC)* and classified by the *Choke Detection Controller (CDC)*. The error instance is then recorded in the *Choke Error Table (CET)*, using an *Error ID (EID)*, for future references. Next, in the *choke error correction* stage, 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) stage. The correction stage incurs large number of penalty cycles, which is equal to the number of pipestages in the processor [8]. Therefore, to avoid the correction flow for repeated errors, the *choke error avoidance* stage is introduced. Earlier researches have shown that errant instructions tend to repeat their behavior owing to commonality in their sequencing and operand values [15, 19]. Trident exploits this intriguing circuit-architectural property. In the avoidance stage, 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. In the next section, we discuss the composition of the EID.

3.3 Error ID (EID)

The EID is a combination of the factors governing an error. The EID comprises the initializing and sensitizing vector [19], the operand sizes of the vectors [2], the class of error (described in Section 3.1) and the errant pipestage. To determine the size of the operands, we simply determine the position of the leftmost set bit. 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). 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 3.4.4. In the next section, we elaborately describe each of the hardware components and their roles in each stage of the design flow.

3.4 Components of Trident

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

3.4.1 Choke Error Table (CET)

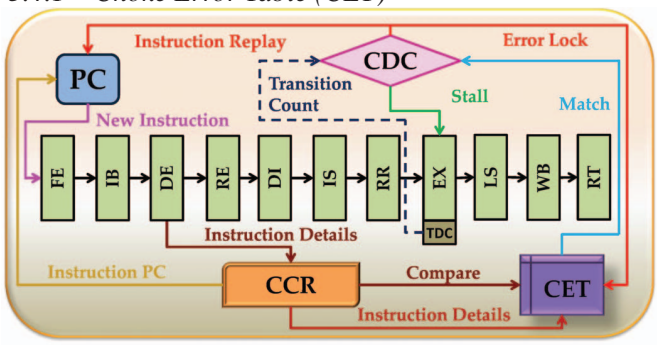
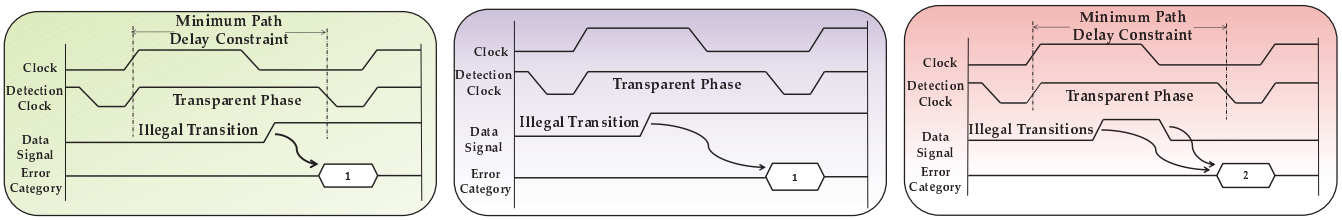


Figure 2: *Design blocks and data flow of Trident. The EX (Execution) pipestage is under scrutiny.*



(a) *Min. Error* **(b)** *Max. Error* **(c)** *Max.-Min. Error*

Figure 3: *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.*

The CET is used to record the error instances, encountered in the choke error detection stage, in the form of EIDs. The table is structured in the form of a Random Access Memory (RAM). In the choke error avoidance stage, the details (discussed in Section 3.3) corresponding to the latest instruction in the CCR are compared against the EIDs. In case there is a match, the CET intimates the class and pipestages of the error to the CDC, for appropriate measures. We consider the CET size to be 128 entries. A comparison among the sizes is left out of this paper, due to the lack of space. If the CET fills up and a new entry is to be made, Pseudo-LRU (Least Recently Used) policy is followed.

3.4.2 Transition Detector and Counter (TDC)

TDCs work only in the choke error detection stage. Every pipestage, between decode (DE) and writeback (WB),is provided with a TDC. Each TDC comprises a double-edged flip-flop [21], to detect both rising and falling transitions. The TDC is controlled by a detection clock, similar to the one described in [6]. 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 in a single clock cycle. When deactivated, the TDC feeds the count to the CDC, for classification. Any transition during this small interval is not flagged as illegal.

3.4.3 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. In the detection stage, it provides the instruction details for the EID. In the *choke error avoidance* stage, it provides the details for comparison to the EID. Further, in the *choke error correction* stage, it provides the PC with the errant instruction address for instruction replay.

3.4.4 Choke Detection Controller (CDC)

This component spearheads the entire design flow of Trident. In the *choke error detection* stage, 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 stage*, where it performs a pipeline flush and indicates the PC to perform an instruction replay. In the *choke error avoidance* stage, the CDC inserts the necessary number of stall cycles (as discussed next) based on the error class information provided by the CET.

Choke Error Avoidance Mechanism: Figures 3a and 3b show the two varieties of SE. For Figure 3a, the transition is early, corrupting the previous instruction results. For Figure 3b, previous instruction result is latched erroneously. Hence, in both these cases, a single stall cycle is required after the instruction that was latched out of the errant stage, to prevent the error. In the first case, the stall cycle ensures that the data latched at the end of the pipestage is not corrupted, due to the minimum timing violation, for one clock period. In the latter case, the stall cycle allows an additional clock period to complete the execution and latch the correct data at the end of the pipestage. Contrary to an SE, a CE causes a chain of data corruptions, shown in Figure 3c. 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. This 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 stages.

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

4. METHODOLOGY

Figure 4 portrays our cross-layer design methodology. In this section, we describe each layer in detail.

4.1 Device Layer

In this layer, we focus on determining the effects of voltage scaling and PV on the gate delays. We use the VARIUS [16] and VARIUS-NTV [12] models, to estimate the effects of PV on the delays of basic logic gates at STC and NTC, respectively. The delay values obtained from HSPICE simulations (discussed in Section 2.3) are used for timing analysis of the circuit, described in Section 4.3.

4.2 Architecture Layer

In this layer of design, we simulate six SPEC CPU2000 benchmarks [10], using the FabScalar infrastructure [4], to generate the input vectors for the synthesized EX stage described in Section 2.3. Further, we augment the EX stage RTL with the *Trident* design components described in Section 3.4. 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 4.3).

4.3 Circuit Layer

In this layer of design flow, we perform the circuit synthesis and the timing analysis. First, the augmented EX stage is synthesized using Synopsys Design Compiler [5] and the

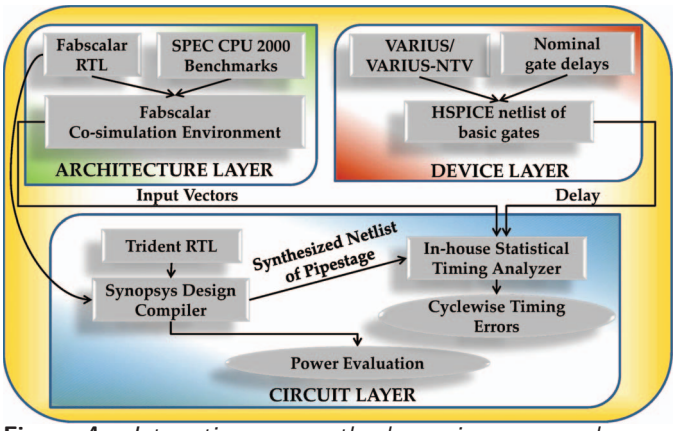


Figure 4: *Interaction among the layers in our cross-layer methodology.*

NanGate library as described in Section 2.3. Next, we conduct a statistical timing analysis with our in-house tool. The tool accepts the synthesized netlist, the input vectors and the logic gate delay values as inputs and generates a cycle-wise sensitized path delay report. We incorporate the effects of PV in the logic gate delay values to emulate the effects of choke points. Finally, we use the path delay report from the tool to analyze the timing violations. We also use Cadence SoC Encounter [3] to place and route the design, and thereby calculate the overall area, wiring and power overheads.

5. EXPERIMENTAL RESULTS

In this section, we evaluate the efficacy of *Trident*. We compare our technique to two timing error detection and mitigation techniques, described in Section 5.1. In Section 5.2, we present the distribution of choke error classes across the benchmarks as detected by *Trident*. In Sections 5.3 and 5.4, we analyze the performance and energy efficiency gains of *Trident*, respectively. Finally, in Section 5.5, we present the overheads associated with *Trident*.

5.1 Comparative Schemes

- **Razor:** This technique detects maximum timing errors in combination paths with the use of a shadow latch [8]. Razor employs buffer insertion to avoid minimum timing violations in short delay paths, and has no error prediction mechanism. This scheme is our baseline for comparison.
- **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 buffer insertion to avoid minimum timing errors.
- **Trident:** Our 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 prediction capabilities.

5.2 Error Distribution

Figure 5 shows the distribution of SEs and CEs across different benchmarks. We insert buffers in the short delay paths (as described in [8]) to analyze the effects of choke buffers. In order to account for all the errors, we disable the choke error avoidance stage during this experiment. As the figure shows, about 80% of all the errors are SEs. For a deeper analysis, we have distinguished the SEs into minimum and maximum timing violations. We observe that

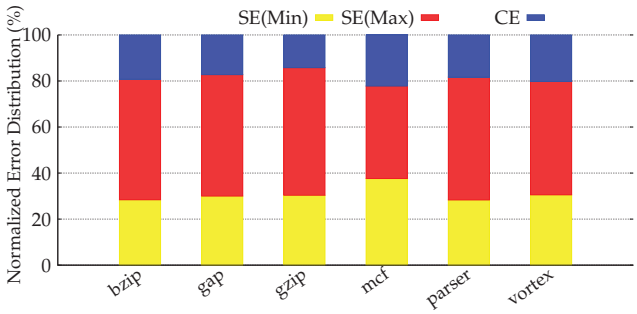


Figure 5: *Distribution of SE and CE for each benchmark. SE are caused by either minimum timing violations [SE(Min)] or maximum timing violations [SE(Max)].*

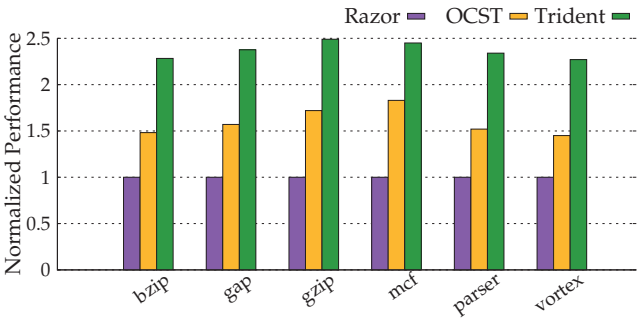


Figure 6: *Performance impact comparison of Trident with Razor and OCST. (Higher is better.)*

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 sections, we discuss the impact of this observation on the performance and energy efficiency of Trident, as compared to the schemes presented in Section 5.1.

5.3 Performance Comparison

Figure 6 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 49.08% 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 and consequently, the execution time of the application. We make an intriguing observation 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.

5.4 Energy Efficiency Comparison

Figure 7 shows the energy-efficiency gain achieved by *Tri-*

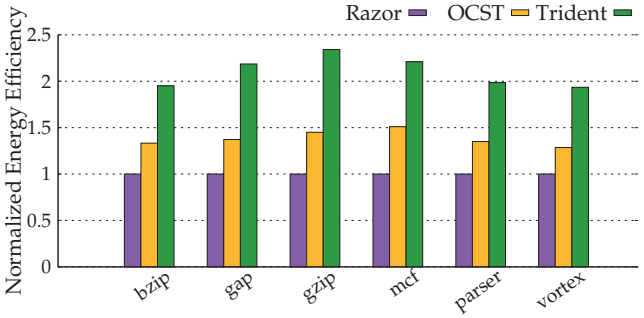


Figure 7: *Energy efficiency comparison of Trident with Razor and OCST. (Higher is better.)*

dent 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, as well as, the reduced overheads (as discussed in Section 5.5). Compared to all the benchmarks, *gzip* shows the maximum gain of 54.62% over OCST and 1.34× over Razor.

5.5 Hardware Overheads

The overheads are calculated after the placement and routing of the EX stage, augmented with the Trident components. The area, power and wiring overheads of Trident, with respect to the unaltered EX stage, are 9.48%, 12.76% and 11.2%, respectively. Compared to the entire pipeline, the area, power and wiring overheads are 0.97%, 1.58% and 1.12%, respectively.

6. RELATED WORK

High PV sensitivity at NTC, and the corresponding impact, has been an active field of research for over a decade. Karpuzcu et al. show how PV sensitivity affects the parameters of an NTC system and can be a potential source of timing errors [11]. They also show that timing error mitigation techniques designed for STC systems cannot be directly adopted for NTC systems. However, only few works have delved into bridging this gap. Tu et al. propose multi-power-mode minimum padding technique to tackle hold time violations in ultra low power systems [18]. Golanbari et al. propose buffer design optimization to address hold time violations at near-threshold voltages (NTV) [9]. Bal et al. propose a dynamic method to detect only critical delay violations caused by choke points [2]. Notably, almost all the techniques rely on some form of buffers to tackle or avoid minimum timing violations at NTC.

To the best of our knowledge, *this paper is the first work that presents choke points as a potential source of minimum timing violations at NTC, as well as, demonstrates the inefficacy of buffer insertion techniques in tackling minimum timing violations in PV affected NTC circuits.*

7. CONCLUSION

In this paper, we explore the impact of minimum timing violations induced by choke points. We also depict the inefficacy of adopting conventional STC timing error detection and mitigation methodologies, at NTC, to tackle these impacts. Buffer insertion technique used in many of these methodologies, to regulate minimum path delays, can be an added predicament to the choke point scenario. To combat this issue, we propose *Trident*, a dynamic and adaptive design paradigm to deal with choke point induced timing errors. *Trident* offers 1.37× performance improvement and 1.1× energy efficiency gain over Razor at NTC, at the cost of marginal hardware overheads.

8. ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grants (CAREER-1253024, CCF-1318826, CNS-1421022, and CNS-1421068). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

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

[2] Bal, A. and others Revamping Timing Error Resilience To Tackle Choke Points at NTC systems. In *Proc. of DATE* (2017), pp. 1020–1025.

[3] Cadence, S. Encounter User Guide.

[4] Choudhary, N. K. and others FabScalar: composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template. In *Proc. of ISCA* (2011), pp. 11–22.

[5] Compiler, D. and others Synopsys. *Inc., see http://www. synopsys. com* (2001).

[6] Das, S. and others RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance. *J. of Solid-State Circ. 44*, 1 (Jan. 2009), 32–48.

[7] De, V. Fine-Grain Power Management in Manycore Processor and System-on-Chip (SoC) Designs. In *Proc. of ICCAD* (2015), pp. 159–164.

[8] Ernst, D. and others Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. In *Proc. of MICRO* (2003), pp. 7–18.

[9] Golanbari, M. S. and others Hold-time violation analysis and fixing in near-threshold region. In *Power Timing, Model. Opt. Sim.* (2016), pp. 50–55.

[10] Henning, J. L. SPEC CPU2000: Measuring CPU performance in the new millennium. *Computer 33*, 7 (2000), 28–35.

[11] Karpuzcu, U. R. and others Coping with Parametric Variation at Near-Threshold Voltages. *IEEE Micro 33*, 4 (2013), 6–14.

[12] Karpuzcu, U. R. and others 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.

[13] NanGate. `http://www.nangate.com/?page_id=2328`.

[14] Paul, B. C. and others Impact of a process variation on nanowire and nanotube device performance. *T. Electron Devices 54*, 9 (2007), 2369.

[15] Roy, S., and Chakraborty, K. Predicting Timing Violations Through Instruction Level Path Sensitization Analysis. In *Proc. of DAC* (2012), pp. 1074–1081.

[16] Sarangi, S. and others VARIUS:A Model of Process Variation and Resulting Timing Errors for Microarchitects. *IEEE Tran. on Semicond. Manufac. 21* (2008), 3 –13.

[17] Sinha, S. and others Exploring sub-20nm FinFET design with predictive technology models. In *Proc. of DAC* (2012), pp. 283–288.

[18] Tu, W.-P. and others Low-power timing closure methodology for ultra-low voltage designs. In *Proc. of ICCAD* (2013), pp. 697–704.

[19] Xin, J., and Joseph, R. Identifying and predicting timing-critical instructions to boost timing speculation. In *Proc. of MICRO* (2011), pp. 128–139.

[20] Ye, R. and others Online clock skew tuning for timing speculation. In *Proc. of ICCAD* (2011), pp. 442–447.

[21] Yu, C.-C., and Chen, K.-T. 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.