TOWARD RELIABLE, SECURE, AND ENERGY-EFFICIENT MULTI-CORE SYSTEM DESIGN

by

Prabal Basu

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

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

Koushik Chakraborty, Ph.D.
Major Professor

Sanghamitra Roy, Ph.D.
Committee Member

Jacob Gunther, Ph.D.
Committee Member

Reyhan Baktur, Ph.D.
Committee Member

Paul Barr, Ph.D.
Committee Member

Richard S. Inouye, Ph.D.
Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

ABSTRACT

Toward Reliable, Secure, and Energy-efficient Multi-core System Design

by

Prabal Basu, Doctor of Philosophy

Utah State University, 2019

Major Professor: Koushik Chakraborty, Ph.D.
Department: Electrical and Computer Engineering

The computing paradigm witnessed a fundamental shift with the uprise of multi-core systems. While pushing the power wall is the primary stimulus, several other conflicting constraints play equally important roles in the design of a multi-core system. For example, ensuring robustness and a secure operation are critical to the deployment of a high-performance system. There are several components of a multi-core system that demand distinct innovations to adhere to all the aforementioned design constraints. This dissertation makes a radical contribution to the following three specific areas of a multi-core system design. First, preserving a reliable operation of a network-on-chip (NoC) components in the presence of a high power supply noise (PSN). The growing NoC power footprint, increase in the transistor current, and high switching speed of the logic devices exacerbate the peak PSN in the NoC power delivery network (PDN). Hence, preserving the power supply integrity in the NoC PDN is critical. A collection of novel flow-control protocols and an adaptive routing algorithm are proposed to mitigate the PSN in NoCs. Second, near-threshold computing (NTC) paradigm is explored in the realm of general-purpose graphics processing units (GPGPU). Two key factors can significantly undermine the efficacy of GPGPUs at NTC: (a) elongated delays at NTC make the GPGPU applications severely sensitive to multi-cycle latency datapaths (MLDs) within the GPGPU pipeline, and (b) process variation (PV) at NTC induces a substantial performance variance. To address these emerging challenges, a dynamic circuit-architectural technique is proposed that can dynamically adjust the degree of parallelization and the speed of the MLDs within each stream core of a GPGPU. Third, an emerging security threat at NTC has been thoroughly analyzed. The timing fault vulnerability of a circuit is shown to increase rapidly, as the operating conditions of the transistor devices shift from traditional super-threshold to near-threshold values. Exploiting this vulnerability, two novel threat models are proposed for NTC that rely on malicious application software to induce timing fault attacks in the underlying NTC hardware. The efficacy of the threat models is evaluated with both simulations, as well as, with off-the-shelf hardware.

(110 pages)

PUBLIC ABSTRACT

Toward Reliable, Secure, and Energy-efficient Multi-core System Design

Prabal Basu

Computer hardware researchers have perennially focussed on improving the performance of computers while stipulating the energy consumption under a strict budget. While several innovations over the years have led to high performance and energy efficient computers, more challenges have also emerged as a fallout. For example, smaller transistor devices in modern multi-core systems are afflicted with several reliability and security concerns, which were inconceivable even a decade ago. Tackling these bottlenecks happens to negatively impact the power and performance of the computers. This dissertation explores novel techniques to gracefully solve some of the pressing challenges of the modern computer design. Specifically, the proposed techniques improve the reliability of on-chip communication fabric under a high power supply noise, increase the energy-efficiency of low-power graphics processing units, and demonstrate an unprecedented security loophole of the low-power computing paradigm through rigorous hardware-based experiments.

To my parents and my brother.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to several persons who have supported me in different capacities throughout the tenure of my Ph.D. First, I would like to thank my major professor Dr. Koushik Chakraborty, and my co-advisor Dr. Sanghamitra Roy, for their continual advice, unabating encouragement, productive feedback, and financial support, that have helped me grow as a researcher over the past few years. Outside of the academia, their hospitality and positive attitude toward life, have had a significant impact on my overall well-being. Second, I would like to thank my Ph.D. committee members Dr. Jacob Gunther, Dr. Reyhan Baktur, and Dr. Paul Barr, for their valuable comments and feedback on my research. I would also like to appreciate the timely support from the staff members of the Electrical and Computer Engineering department. I am thankful to Mary Lee Anderson and Tricia Brandenburg for helping me out with all the academic paperwork; Trent Johnson, Scott Kimber, and Steve Meeker, for the technical support that had immensely facilitated my research and TA jobs. I would also like to thank Anup Kumar Sultania, Sabyasachi Das, Rajat Aggarwal, and Pranay Prakash, for assisting me with my internships at Xilinx and Intel.

I am extremely thankful for my friends and colleagues at the BRIDGE lab, who have played instrumental roles during my Ph.D. years. My heartfelt thanks to Shamik for being a delightful housemate and a partner in many course and research projects; Rajesh for the great support during my maiden research undertaking and subsequent grant proposal works; Chidham for being a fantastic housemate, a fine experimental cook, and my partner in several projects; Hu for his significant contribution to the GPU work; Asmita for her cheerful presence and occasional discourse on research; Aatreyi for her excellent commitment to all the projects that we did together; Pramesh for being an outstanding research partner and for his nonchalant attitude toward life; Tahmoures for his kind, diligent and sophisticated personality; and Sourav for being an appreciative person, an easygoing housemate, and a

humble protégé. I would also like to thank Dean, Manzi, Atif, Kurt, and Trevor, for helping me out in meaningful ways during my stint as a Ph.D. scholar.

Outside of the lab, I am grateful for the presence of many persons in my life. Tarak for being an excellent housemate for several years; Dipanjan for always being there whenever I sought his academic guidance; Tanwir, Sudipta, Anway, and Saptarshi for being so hospitable during my early days in Logan; Subhendu for his invaluable professional guidance and the prolonged discussions on research, industry and life; and Dr. Ravi Gupta and his family for being the kind hosts of the Friday night spiritual gathering over the years. I would also like to thank my friends Padma, Bagchi, Antu, Jana, Avik, Debrup, Sarbajit and Siddhartha, for making my journey pleasurable and amusing.

Last but not least, I would like to express my deepest gratitude to my parents Mita Basu and Pulak Kumar Basu, and my brother Pratip Basu. Without their love, sacrifice, motivation and support, this dissertation would not have been possible.

Prabal Basu

CONTENTS

LIST OF FIGURES

Figure | Page

ACRONYMS

NoC	Network-on-Chip
MPSoC	Multiprocessor System-on-Chip
PSN	Power Supply Noise
PDN	Power Delivery Network
PAF	PSN-Aware Flow-control
PAR	PSN-Aware Routing
DOR	Dimension Order Routing
DBAR	Destination-Based Adapting Routing
DSENT	Design Space Exploration for Network Tool
MCL	Maximum Current Load
FLAP	Flit Acceptance Potential
VC	Virtual Channel
PARSEC	Princeton Application Repository for Shared-Memory Computers
VLSI	Very Large Scale Integration
EDP	Energy Delay Product
STC	Super-Threshold Computing
NTC	Near-Threshold Computing
NTV	Near-Threshold Voltage
GPGPU	General Purpose Graphics Processing Unit
MLD	Multi-cycle Latency Datapath
PV	Process Variation
CU	Compute Unit
TSMC	Taiwan Semiconductor Manufacturing Company
SAS	Self-Adaptive Sprint
SSE	Static Sprint Execution
FO4	FanOut-of-4
FU	Functional Unit

TITAN	Timing Fault Attack at NTC
IoT	Internet of Things
RISC	Reduced Instruction Set Computer
MIPS	Microprocessor without Interlocked Pipelined Stages
ISA	Instruction Set Architecture
PoFF	Point of First Failure
FPGA	Field Programmable Gate Array
DCM	Digital Clock Manager
PR	Partial Reconfiguration
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
HOP	Highest Operating Point
FIFA	Focally Induced Fault Attack
STA	Statistical Timing Analysis
RTL	Register Transfer Level
PTM	Predictive Technology Model
SPICE	Simulation Program with Integrated Circuit Emphasis
ITRS	International Technology Roadmap for Semiconductors

CHAPTER 1
INTRODUCTION

The proliferation of multi-core devices has significantly increased the design complexity of modern computing systems. Such complexity stems from several conflicting, first-order design constraints. For example, ensuring a high system performance under a tight power budget requires a careful cross-layer design, spanning device, circuit, and architecture layers. Moreover, with ever-shrinking transistor features, integrating more devices on a chip has started to raise several reliability concerns in contemporary technology nodes. As a result, maintaining a functionally correct execution at the component-level can potentially undermine the system-level energy efficiency. While low-power system design can address some of these challenges, it is plagued from an inherent security vulnerability due to a low-voltage operation of the transistor devices. This dissertation aims to tackle some of the critical challenges in attaining reliable, energy-efficient, and secure multi-core systems.

One of the major sources of reliability challenges in modern computers is *Power Supply Noise (PSN)*. PSN can cause a sporadic droop in the supply voltage, radically degrading the delay of various on-chip circuit components. A direct impact of PSN is *timing errors*, where the delay of a pipe stage computation exceeds the clock period. Network-on-Chip (NoC)—the emerging de-facto standard for on-chip communication platform [1, 2]—faces unique challenges from PSN, primarily due to the simultaneous switching of transistor devices in lower technology nodes. This dissertation proposes novel proactive approaches, including a slew of flow-control protocols and an adaptive routing algorithm, to mitigate the peak PSN in NoCs, while incurring a minimal performance loss.

Near-Threshold Computing (NTC) is emerging as one of the promising low power computing platforms, promoting an energy-efficient system design [3–5]. An NTC device sets its supply voltage close to its threshold voltage, dramatically reducing the energy consumption. However, the reduced operating voltage at NTC also leads to a significant performance loss in a uni-core processor. In order to maintain the traditional super-threshold computing (STC) performance at NTC, application parallelism can be exploited, thus making GPUs an excellent platform for reaping the benefits of NTC. This dissertation explores several predicaments of the NTC operation of a GPU, and proposes a holistic circuit-architectural solution to promote an energy-efficient NTC execution of general-purpose GPU applications.

Moving from the traditional STC to NTC alters many of the device characteristics, jeopardizing trustworthy computing. For example, a slight variation in the supply voltage impacts the circuit delay characteristics significantly, enabling copious security loopholes at NTC [3]. While many recent works have explored challenges and opportunities of reliable computing at NTC, the *security vulnerabilities* spawning from such low power computing frameworks have received only marginal attention. This dissertation establishes a platform to reason about trustworthy computing on a system built with circuits and devices operating at the near-threshold regime.

1.1 Contributions of This Dissertation

The works presented in this dissertation have been published in several conference proceedings and journal articles, including 2016 IEEE/ACM Design, Automation and Test in Europe (DATE), 2016 IEEE/ACM Design Automation Conference (DAC), 2017 IEEE Transactions on Very Large Scale Integration Systems (TVLSI), 2018 IEEE Embedded Systems Letters (ESL), and 2018 IEEE Transactions on Emerging Topics in Computing (TETC). Details of the publications are listed below:

1.1.1 Conference Papers

- SwiftGPU: Fostering Energy Efficiency in a Near-Threshold GPU Through a Tactical Performance Boost. Prabal Basu, Hu Chen, Shamik Saha, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM Design Automation Conference (DAC)*, 2016.

- PRADA: Combating Voltage Noise in the NoC Power Supply Through Flow-Control and Routing Algorithms. Prabal Basu, Rajesh JS, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM Design Automation and Test in Europe (DATE)*, 2016.

1.1.2 Journal Papers

- 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 (TETC)*, 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 (ESL)*, vol. 10, issue. 4, pp. 115-118, 2018.

- IcoNoClast: Tackling Voltage Noise in the NoC Power Supply Through Flow-Control and Routing Algorithms. Prabal Basu, Rajesh Jayashankara Shridevi, Koushik Chakraborty and Sanghamitra Roy. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 25, issue 7, pp. 2035-2044, 2017.

CHAPTER 2

LITERATURE REVIEW

This chapter presents an extensive literature survey on the existing research efforts that are related to this dissertation. The pertinent works include analysis and mitigation of PSN in various system-on-chip (SoC) components, circuit and architecture-level techniques that improve the performance and energy-efficiency of general purpose GPUs, and security vulnerabilities in low-power devices. In order to systematically study the important works in these areas, this chapter is organized in the following sections. Section 2.1 discusses the works on PSN mitigation strategies, Section 2.2 describes several existing optimization techniques to improve the power-performance of GPUs, and Section 2.3 outlines the state-of-the-art works on diverse hardware security attacks and their countermeasures.

2.1 Study of PSN in SoC Components

Several previous research efforts study the significance of PSN on the chip power performance. The combined effect of the aggravating transient current and accelerated on-chip component density has raised severe reliability concerns. Work related to reducing peak voltage noise can be categorized in three domains: 1) characterizing and mitigating PSN in microprocessors (Section 2.1.1); 2) understanding voltage noise in NoC (Section 2.1.2); and 3) flow-control and routing techniques in the NoC (Section 2.1.3).

2.1.1 Characterizing and Mitigating Peak Noise in Microprocessors

- **VRSync [6]:** The effects of chip-wide activity fluctuation, caused by global synchronization in multi-threaded applications, can overwhelm the effects of core-level workload variability. Based on this observation, Miller et al. developed a methodology that uses emergency-aware scheduling policies to reduce the slope of load fluctuations, eliminating voltage emergencies.

- **Orchestrator** [7]**:** Xing et al. observed that multiple threads sharing the same function body exhibit similar power activity. As a result, voltage emergencies can be mitigated intelligently by relocating the threads among the cores. Their proposed technique Orchestrator maximally leverages thread diversity based on inter-core voltage interaction, thus obviating voltage droop synergy among cores.

- **Sharma et al.** [8]**:** Voltage emergencies are correlated with specific microarchitectural event like cache misses. The authors of this work proposed an event-guided dynamic technique that triggers a droop-avoidance mechanism when emergency-prone events recurs.

None of these works explore the impact of interconnect fabric (NoC) on power supply noise. With the growing power footprint of NoC in a multi-core system, reduction of peak supply noise in NoCs is critical for reliable and energy-efficient computing.

2.1.2 Understanding Voltage Noise in NoCs

- **Nostrum** [9]**:** Penolaazi and Jantsch's high level power model for the *Nostrum* NoC was one of the earliest efforts to develop an empirical function to accurately estimate power fluctuations for an NoC load. A system simulation with their power model can run up to 500 times faster than with Synopsys Power Compiler for a 4X4 NoC.

- **Dahir et al.** [10]**:** The authors developed a tool for measuring the voltage drop in NoC routers due to communication workloads. The tool comprises an NoC simulator, a fast power grid model, an on-chip link model, and a microarchitectural power model for the routers. The model used in the tool has a mean relative error of 4.7% with respect to SPICE simulation.

- **Dahir et al.** [11]**:** A novel application to core mapping strategy was proposed to create a balanced activity distribution across the whole chip. Consequently, a significant reduction in power supply noise was obtained with minimal energy overhead.

Contrary to the aforementioned works, this dissertation explores proactive, architectural techniques in NoCs to lower the peak voltage noise due to NoC activity.

2.1.3 Flow-control and Routing Techniques:

- **Elastic Buffer** [12]**:** In order to reduce the area and power footprints, Michelogiannakis et al. developed an elastic buffer (EB) flow control which adds simple control logic in the channels to use pipeline flip-flops as EBs with two storage locations. This, in turn, improved the throughput of the network.

- **Kang et al.** [13]**:** A fault-tolerant flow-control protocol is proposed to tackle soft-errors in NoCs. The flow-control protocol recovers errors by the flit re-transmission that ensures an error-free transmission on a flit-basis with dynamic packet fragmentation. 97% error coverage was obtained with the use of the proposed technique.

- **CATRA** [14]**:** A congestion aware routing algorithm was proposed by utilizing both local and non-local network information thus choosing the optimal path for forwarding a packet. A history based congestion detection metric was used along with a distributed propagation system to ensure an accurate and efficient estimation of the on-chip packet congestion.

- **Fick et al.** [15]**:** A fault-tolerant routing algorithm was proposed for 2D-mesh and 2D-torus networks that overcomes a large number of faults in a fine-grained fault model, offering an astounding 99.99% reliability, on an average.

Unlike these works, this dissertation explores the use of flow-control and routing algorithms in peak noise mitigation in NoC while improving the NoC energy efficiency.

2.2 Improving GPU Energy-efficiency

Several existing works that aim at increasing the GPU energy efficiency at the super-threshold region, can be broadly classified into two categories: 1) throttling the hardware to save energy; and 2) refining thread-scheduling and using novel memory design techniques to increase the utilization of GPUs. On the other hand, several works have delved into the principles of the near-threshold region, as well as, its opportunity in micro-architecture.

2.2.1 Hardware Throttling Mechanism to Improve GPU Energy-efficiency

- **Wang et al.** [16]**:** The authors proposed a run-time power gating technique for GPU caches to save leakage energy. The L1 and L2 caches go to a low-leakage sleep mode when there is no ready thread and no memory request, respectively. This policy is architecturally invisible and incurs a minimal overhead while achieving 54% energy saving, on an average, for several CUDA benchmarks.

- **Lee et al.** [17]**:** The work demonstrated that, under a power envelope, adjusting the number of GPU cores and the voltage/frequency of cores and/or on-chip interconnects/caches can improve the throughput. Moreover, dynamically scaling the number of operating cores and the voltages/frequencies of both cores and on-chip interconnects/caches at runtime can further improve the throughput of GPU applications.

- **GreenGPU** [18]**:** The authors proposed an energy management framework for GPU-CPU heterogeneous architectures. The proposed strategy dynamically distributes the workloads to GPU and CPU so as to ensure that both the devices can finish their executions approximately at the same time. Consequently, the energy consumption for idling and waiting for the slower device to finish its execution is reduced significantly. Furthermore, based on the utilizations of the CPU and GPU, the respective core frequency and voltage are adjusted to save energy.

2.2.2 Improving GPU Utilization

- **Narasiman et al.** [19]**:** The authors claimed that the GPU resources are grossly under-utilized. They suggested a large-warp micro-architecture to alleviate the performance degradation due to branch divergence. Also, a two-level warp scheduling was proposed to prevent all warps from stalling together.

- **Wang et al.** [20]**:** Exploiting a close cooperation between compiler-directed data placement and hardware-assisted runtime adaptation, this work uses a phase change memory based hybrid memory design, to improve the utilization of the GPUs.

2.2.3 Challenges and Opportunities of NTC

- **Pinckney et al.** [21]**:** This work explored how the cessation of Dennard scaling can be tackled by a near-threshold voltage operation of a chip multiprocessor while utilizing the inherent Parallelism of the applications. Considering the parallelization overhead, an NTC operation provides 4X improvement in the chip multiprocessor performance across 6 commercial technology nodes.

- **Dreslinski et al.** [3]**:** Several key challenges of the NTC operation of a circuit are discussed. For example, 10X performance loss, 5X increase in performance variation, and 5 orders of magnitude increase in functional failure rate of memory, all with respect to traditional super-threshold operation. Additionally, several cross-layer solutions are proposed to gracefully tackle these problems, leading to an improved energy-efficient operation of NTC circuits and systems.

- **Marković et al.** [5]**:** The realm of the NTC operation of a circuit was explored with variations in supply voltage and transistor sizing. The authors introduced a pass-transistor based logic family with only sub-threshold leakage while operating at the near-threshold region. Finally, the use of ultra-low power design technique was demonstrated in the chip synthesis methodology.

- **Hsu et al.** [22]**:** The authors proposed a reconfigurable single instruction multiple data vector permutation engine that can work at the NTC region while tolerating process variation. The register file circuit optimizations for ultra-low voltage operation enabled robust functionality measured down to 280mV consuming 109 micro-watt at 16.8MHz, with peak energy efficiency of 154GOPS/W (9X higher than nominal).

Compared to the aforementioned research efforts, this dissertation explores the challenges and performance/energy benefits of near-threshold execution of GPUs.

2.3 Study of Hardware-oriented Security Vulnerability

Works on hardware security most relevant to this dissertation can be categorized as 1) fault attacks (Section 2.3.1), 2) side-channel attacks (Section 2.3.2), 3) hardware trojan attacks (Section 2.3.3), and 4) security threats in low-power systems (Section 2.3.4).

2.3.1 Fault Attacks

Fault Attacks (FA) leak secure and confidential information by inducing faults into the circuits' computations, or by modifying the circuit environment in order to change its expected behavior. Some of the popular means of FAs are chip undervolting, overclocking, electromagnetic pulse, laser beam and high temperature. Researchers have broadly subcategorized FAs into *safe error*, *algorithm modification* and *differential fault analysis (DFA)*. Safe error attacks distinguish between normal and abnormal behaviors of the chip to retrieve sensitive information [23]. Balasch et al. exemplified algorithm modification, by replacing the instructions executed by a microcontroller [24]. On the other hand, DFA extracts the cryptographic keys by comparing the original and faulty ciphertexts [25]. Recently, Chong Kim improvised the DFA on AES key schedule [26]. Tangil et al. proposed *Alterdroid*, a DFA of obfuscated smartphone malware [27]. Sakiyama et al. presented an information-theoretic mechanism for an optimal DFA [28]. Zussa et al. thoroughly investigated timing constraints violations to achieve fault injection [29]. Mukherjee et al. [30] and Yuce et al. [31] estimated the vulnerabilities of hardware cryptosystems, and proposed some metrics to accurately measure the vulnerability from fault attacks. This dissertation demonstrates the aggravated security threats of the low-power circuits, using rigorous hardware experiments.

2.3.2 Side-channel Attacks

Side channel attacks (SCA) exploit the information leakage from the physical implementation of a cryptographic hardware, for example, electromagnetic emission, power consumption and computation time. Kocher et al. were the early pioneers of side channel analysis who demonstrated a strong correlation between the power consumption measurements, and the cryptographic operations [32]. Basin et al. combined the side-channel analysis with an information theoretic model, to quantify the information revealed to an attacker [33]. While Liu et al. presented the efficacy of the SCA in last-level caches [34], Wang et al. proposed new cache design paradigm to counter such cache-based SCA [35]. Hund et al. demonstrated the potency of SCA against the memory management system, to gain privileged address space information [36]. Luo et al. recently analyzed the SCA vulnerabilities of the GPUs, in the context of secure cryptographic implementation [37]. Although, the research in SCA has matured over the last decade, this dissertation is the first work exploring the exclusive security threats at the near-threshold regime.

2.3.3 Hardware Trojan Attacks

Malicious modification of electronic circuits, also known as the *Hardware Trojan*, is a well researched area in hardware security. Tehranipor et al. were one of the first to systematically classify various hardware trojans [38]. Using a cryptanalysis based IC fingerprinting methodology, Agarwal et al. proposed a trojan detection mechanism [39]. Bhunia et al. thoroughly analyzed the threats from hardware trojan attacks, presented different attack models, and proposed various forms of protection approaches [40]. Among the more recent works, Salmani et al. improved the trojan activation time, using a dummy scan flip-flop insertion method [41]. Narasiman et al. improved the detection sensitivity of small trojans with a side channel analysis [42]. Exploiting post-silicon multimodal thermal and power characterization techniques, Hu et al. proposed an accurate trojan detection mechanism [43]. The proposed threat models in this dissertation assume a trojan free trusted hardware, while exploiting an artifact in the hardware layer.

2.3.4 Security Threats in Low-power Systems

Researchers have explored the security vulnerabilities of low-power systems, for example, sensor networks [44, 45], and implantable medical devices [46, 47], most of which are evident due to the unencrypted wireless interfaces in the communication level. The mitigation is centered around developing resource constrained encryption techniques and security protocols [45,48]. However, the circuit/microarchitectural security vulnerabilities that may arise due to the low-power operation of these systems, have not been explored. This dissertation focuses on uncovering such hardware level vulnerabilities for the low-power devices, and proposes novel threat models that exploit the vulnerabilities to inflict end-user harms.

CHAPTER 3

TACKLING VOLTAGE NOISE IN THE NOC POWER SUPPLY THROUGH FLOW-CONTROL AND ROUTING ALGORITHMS

3.1 Background and Contributions of This Work

Supply voltage integrity is a growing concern in modern multiprocessor system-on-chips (MPSoCs). The varying current demand due to the simultaneous switching of the logic devices, creates a noise in the power delivery network (PDN), resulting in a drop in the effective supply voltage. This power supply noise (PSN) has a detrimental effect on the performance, reliability and energy efficiency of various system components. Unfortunately, with the shrinking technology nodes, this problem is poised to grow significantly due to the decreasing feature size, high device density and interaction among many connected components. As current and upcoming MPSoCs are embracing Network-on-Chips (NoCs) as their de-facto standard for on-chip communication, PSN will negatively impact fault-free communication on them.

Modern day NoCs can consume a significant fraction of the total chip power (∼36% in the 80-tile TeraFLOPS at 65-nm [1]). Moreover, researchers are dedicating an independent power-grid network for the NoC to enable efficient power management [49]. Consequently, conventional techniques (e.g., [6, 8]) to tackle the PSN in cores have little impact on the noise in an NoC PDN. Collectively, these trends make it imperative to control the PSN in an NoC PDN, to ensure fault free and energy efficient communication.

In this chapter, a strong correlation between traffic patterns and the peak PSN in the NoC is established. Using a rigorous cross-layer analysis, it is demonstrated that simultaneous and sudden rise in traffic loads within proximal regions in an NoC can lead to a significant voltage noise. Subsequently, it is shown that existing NoC flow-control protocols and congestion aware routing algorithms are unable to mitigate the PSN problem effectively.

For example, a representative congestion aware routing scheme, called Destination-Based Adapting Routing (DBAR [50]), shows negligible 0.1-1% average PSN improvements in real workloads, over a deterministic dimension order routing.

Guided by this cross-layer insight, a combination of static design time and low-complexity runtime approaches are explored that can efficiently mitigate voltage noise in an NoC PDN. The proposed flow-control protocol intermittently allows high and low flit receptions within a single NoC component, while systematically applying a hierarchical approach for scalability. To further improve voltage noise characteristics, a flow-control cognizant adaptive routing is explored, that proactively disperses the flit routes. Collectively, the proposed mechanisms incur marginal circuit level implementation overheads, while promoting energy efficient communication over the NoC by dampening voltage noise on its PDN.

The specific contributions in this chapter are discussed next.

3.1.1 Contributions

- The trends in interconnect circuit parameters and their impact on the peak PSN in NoCs are presented (Section 3.2.4 and 3.2.4).
- The correlation between the peak PSN and the NoC router activity is analyzed (Section 3.2.4).
- It is shown that congestion aware routing algorithms are ineffective in mitigating the peak PSN (Section 3.5.2).
- A couple of runtime solutions, collectively referred to as IcoNoClast, are proposed to mitigate PSN in NoCs. IcoNoClast comprises a novel *PSN-Aware Flow-control (PAF)* protocol and an adaptive *PSN-Aware Routing (PAR)* algorithm (Section 3.3).
- The best scheme of IcoNoClast can reduce the regional peak PSN by ∼15% and improve the energy efficiency by ∼12% compared to a representative routing scheme (DBAR), with a nominal 4.1% average performance overhead and marginal area/power overheads (Section 3.5).

3.2 Motivation

In this section, the impact of power supply noise in NoCs is discussed. Section 3.2.1 discusses the components of supply noise, Section 3.2.2 briefly outlines the supply noise estimation methodology, Section 3.2.3 demonstrates the inefficacy of the core noise mitigation techniques in reducing the NoC PSN, Section 3.2.4 presents the PSN trends, and finally Section 3.2.6 motivates the need for a PSN-aware flow-control protocol.

3.2.1 Power Supply Noise (PSN)

The sources of voltage noise in a PDN are (a) resistive drop (IR) and (b) inductive drop ($L\frac{\Delta i}{\Delta t}$). Voltage drop across the resistances of the power delivery wires causes IR drop, which is proportional to the current (I) in the circuit. Inductive drop, on the other hand, is caused by the wire inductance (L) of the power grid and is proportional to the rate of change of current through the inductance.

3.2.2 PSN Estimation Methodology

Accurate estimation of the peak PSN presents methodological challenges. These challenges stem from cycle-accurate tracking of the pipeline activities of an NoC router, and evaluating its effect on the PDN. The rigorous cross-layer methodology to tackle these challenges are outlined next (details in Section 3.4). First, as per the emerging trends, a dedicated standalone PDN for the NoC is considered [49] to discount the effect of the processing cores' activities on the NoC PDN. Second, energy consumptions by different pipeline stages of a router are collected using the DSENT tool [51]. Third, the interconnect R,L,C parameters for various technology nodes are estimated from the ITRS report [52]. Fourth, a recently proposed MATLAB based PSN tool [10] is converted to C++ and integrated with the Booksim2.0 NoC architectural simulator [53]. Using the router activity traces generated by running real workloads, interconnect circuit parameters and router pipeline energies, the integrated PSN tool derives cycle accurate supply noise statistics of the NoC.

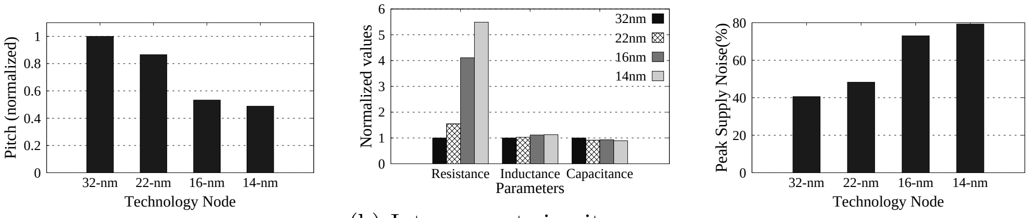
Benchmarks	**Correlation Co-efficient for 5% Droop**	**Correlation Co-efficient for 10% Droop**
Barnes	0.018	0.022
Raytrace	0	-0.140
Radiosity	0.070	-0.054
Water_sp	0.063	0.112

Table 3.1: Correlation between high core droop events and high NoC droop events.

3.2.3 Limitation of Noise Resilient Microprocessor Schemes in Tackling the NoC PSN

Noise resilience schemes in microprocessors are ineffective in tackling the NoC PSN problem due to several key factors. First, researchers are dedicating an independent power-grid network to NoCs for flexible power management [49]. Consequently, conventional techniques to tackle the PSN in cores will have little impact on the noise in an NoC PDN. Second, specific control flow and microarchitectural event sequences may cause voltage emergencies in the microprocessor pipelines (e.g., a high power instruction followed by a low power one [54]). As a result, pipeline techniques are designed to alter such event sequences dynamically. Pipeline droop mitigation techniques can only alleviate voltage noise in the NoC if there is a strong correlation between voltage droop events on the NoC and similar events at the core pipeline.

Table 3.1 presents the experimental data to demonstrate that in reality, there is negligible correlation between large voltage droops at the core pipeline and large voltage droops at the NoC. For example, assuming a 10% droop as a threshold for high droop, a low 0.02 correlation is found for the benchmark *Barnes*. This data is collected using a 64-core Intel Xeon (X5550) processor simulated using the Sniper simulator [55], running representative benchmarks. The occurrence timestamps of processor and NoC droops are carefully analyzed to estimate their correlation. This intriguing result stems from fundamental differences in the activity characteristics within a processor pipeline and the NoC. *Unlike a processor pipeline, the NoC activity is driven by traffic load, rather than individual power footprints of interleaved instructions.* Consequently, there is a critical need to explore techniques to



(a) Interconnect pitch.

(b) Interconnect circuit parameters per unit length.

(c) Peak supply noise.

Fig. 3.1: Trends of interconnect pitch [Figure 3.1(a)], circuit parameters (R,L,C) [Figure 3.1(b)] and peak voltage noise due to technology scaling [Figure 3.1(c)]. Results of Figure 3.1(a) and 3.1(b) are normalized to the corresponding 32-nm technology values.

mitigate voltage droop in the NoC.

3.2.4 Results of PSN Trends

In this section, the impact of technology scaling on the PSN is investigated.

Impact of Scaling on Interconnect Parameters

Figure 3.1(a) shows gradual reduction in the global interconnect pitch (M9 global wire [56]) with technology scaling (ITRS 2013 [57]). As the interconnect width decreases, the resistance per unit length of the metal layer increases rapidly. Figure 3.1(b) shows the trend of R,L,C parameters at smaller technology nodes. The NoC is expected to grow in its area footprint to enable communication among an increasing pool of on-chip components. As a collective impact of these trends, *communication through an NoC will entail an increasing current* for charging/discharging a potentially growing pool of charge reservoirs (device and wire capacitance), while incurring a greater resistive drop due to the increased interconnect resistance.

Impact of Scaling on the Peak Supply Noise

Simultaneous switching of transistor devices causes a large variation in current, leading to a high inductive noise due to the wire inductance. Such a high inductive noise is responsible for the intermittent peaks in the cycle-wise noise profile of a system. As the power supply voltage scales down, the peak supply noise (as a percentage of the supply voltage)

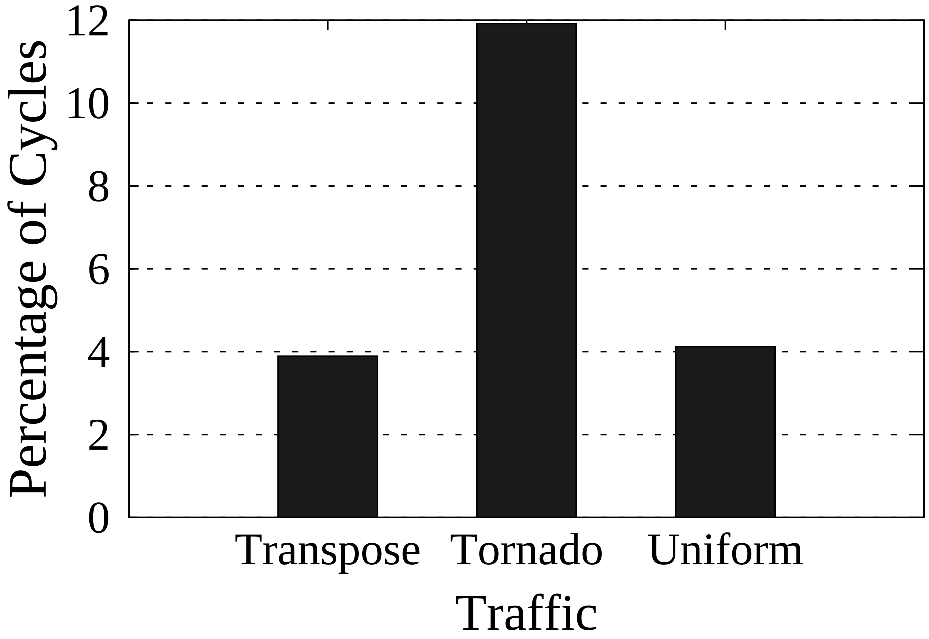


Fig. 3.2: Measure of high load change in consecutive cycles, for the most exercised router (DBAR). Bars represent the percentage of total cycles, when the difference in the number of flits served in two consecutive cycles is 3 or 4 or 5. A higher percentage indicates a more bursty nature of the traffic pattern.

increases at lower technology nodes. Figure 3.1(c) shows that, in an 8X8 NoC running uniform-random synthetic traffic, the peak noise increases from 40% of the supply voltage at the 32-nm technology node to about 80% of the supply voltage at the 14-nm technology node, if the power distribution strategy remains unchanged.

Peak Supply Noise vs. Router Activity

Figure 3.2 shows the traffic load change (measured as a difference in the number of incoming flits in two consecutive cycles) on the most exercised router (DBAR) of the NoC, for a few representative traffic patterns. The y-axis denotes the total percentage of cycles that have a difference of x (x can be 3, 4 or 5) flits, served in two consecutive cycles. For example, for the *tornado* traffic pattern, the most exercised router spends nearly 12% of the execution time with a high change in the number of flits served in consecutive cycles, denoting sudden bursts of data. Figure 3.2 also implies that for a considerable fraction of the total cycles, the router activity changes dramatically, resulting in a large PSN. Figure

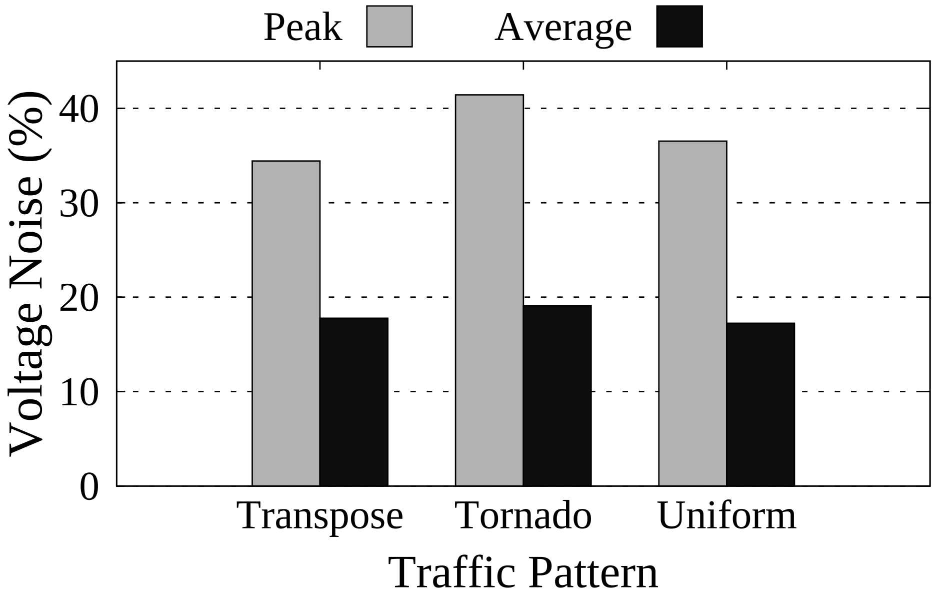


Fig. 3.3: Peak and average voltage noise for the most exercised router (DBAR) at the 32-nm technology node.

3.3 shows the peak and average voltage noise characteristics for various synthetic traffic patterns, by employing a representative congestion aware routing algorithm—DBAR. The peak voltage noise data indicates a correlation of the PSN with the bursty nature of synthetic traffic. In this case, *tornado* traffic pattern suffers with high PSN due to the high percentage of load change per cycle, compared to other traffic patterns. The average PSN across traffic patterns is fairly consistent, as the router undergoes nominal change in router activity for most of the total execution time.

3.2.5 Inefficacy of Existing Routing Schemes in Mitigating PSN

Many congestion-aware routing schemes choose either XY or YX routing paths based on the congestion in the network. Figure 3.4 illustrates a concrete example of why such a strategy is ineffective for PSN mitigation. In this example, nodes 0,1,2 and 6, inject flits to the destination node 5, in a network employing wormhole flow control. Cases 1 and 2 use XY and YX routes, respectively. It can be seen that, although the routing paths are different, the per cycle maximum activity (e.g., flit reception) in region A remains unchanged if all flits are delivered in the same cycle. The possibility of multiple incoming

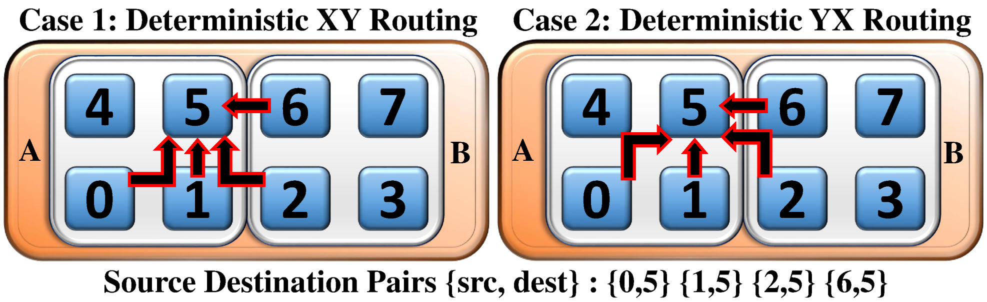

Case 1: Deterministic XY Routing
Case 2: Deterministic YX Routing
4
5
6
7
0
1
2
3
A
B
Source Destination Pairs {src, dest} : {0,5} {1,5} {2,5} {6,5}

Fig. 3.4: An illustrative example revealing the inefficacy of routing algorithms in alleviating regional PSN.

flits in the same cycles is presented in Section 3.2.4. Hence, both the cases incur similar PSN in region A. Since existing routing schemes do not consider the temporal nature of flit delivery to a router, they are inefficient in mitigating PSN. A concrete quantitative evidence of this negligible impact of congestion aware routing schemes on the PSN is presented in Section 3.5.2. The proposed schemes monitor and adaptively alter the temporal nature of flit delivery to effectively reduce the PSN in NoCs.

3.2.6 Impact of Flow-control on the NoC PSN

Existing flow-control protocols (e.g., wormhole) are also unable to mitigate PSN in NoCs. Simultaneous activity in proximal routers causes a high switching current to be drawn from the NoC PDN, leading to a large drop in the supply voltage. State-of-the-art flow-control policies govern the flit propagation in the NoC using a credit system, which keeps track of the buffer availability in downstream routers. In a 2D mesh NoC topology, a router can potentially receive a maximum of 4 flits (not considering the injection/ejection port) in a cycle. As the availability of credits of a router is visible to the adjacent nodes, proximal routers can potentially receive a large number of flits in the same cycle causing a local spike in the PSN profile of the NoC. As the range of possible credit utilization goes up in high radix topologies (e.g., butterfly), the peak PSN is substantially exacerbated.

Inspired by these circuit-architectural insights, novel flow-control and routing algorithms are developed that work in harmony to mitigate noise in an NoC PDN.

3.3 PSN Aware Runtime Adaptations

In this section, a collection of novel *PSN-Aware Flow-control (PAF)* protocols and an adaptive *PSN-Aware Routing (PAR)* algorithm, collectively known as IcoNoClast, is presented. IcoNoClast aims to dampen high simultaneous current loads in proximal regions, by dynamically altering their respective *flit acceptance potentials* and proactively dispersing the flit routes in the network. The design challenges are outlined in Section 3.3.1, before presenting the PAF and PAR techniques in detail in Section 3.3.2 and 3.3.3, respectively. Section 3.3.4 summarizes the advantages of the proposed techniques, and Section 3.3.5 concludes with the implementation details.

3.3.1 Design Challenges

(a) Performance impact: Run-time adaptations to mitigate PSN should have a low performance overhead.

(b) Starvation avoidance: Throttling the *flit acceptance potential* of a router can create buffer back-pressure in the upstream routers. Under a high flit injection rate, the back-pressure can grow so large that it may lead to a starvation. It is important to guarantee freedom from starvation in IcoNoClast.

(c) Scalability: A PSN improvement technique should scale with the size of the communication fabric. It is imperative to minimize its implementation overhead so as to sustain its efficacy in future exascale computing.

3.3.2 Design of PAF

The design of PAF involves a hierarchical approach to dictate the *Maximum Current Load (MCL)*[1] across the NoC, while ensuring a minimal performance impact. Section 3.3.2 outlines an overview of PAF, Section 3.3.2 presents an illustrative example, and Section 3.3.2 discusses the optimizations of PAF.

[1] MCL of an integrated circuit in an epoch (few cycles) is defined as the highest possible amount of current that the circuit can draw from the power supply, in that epoch.

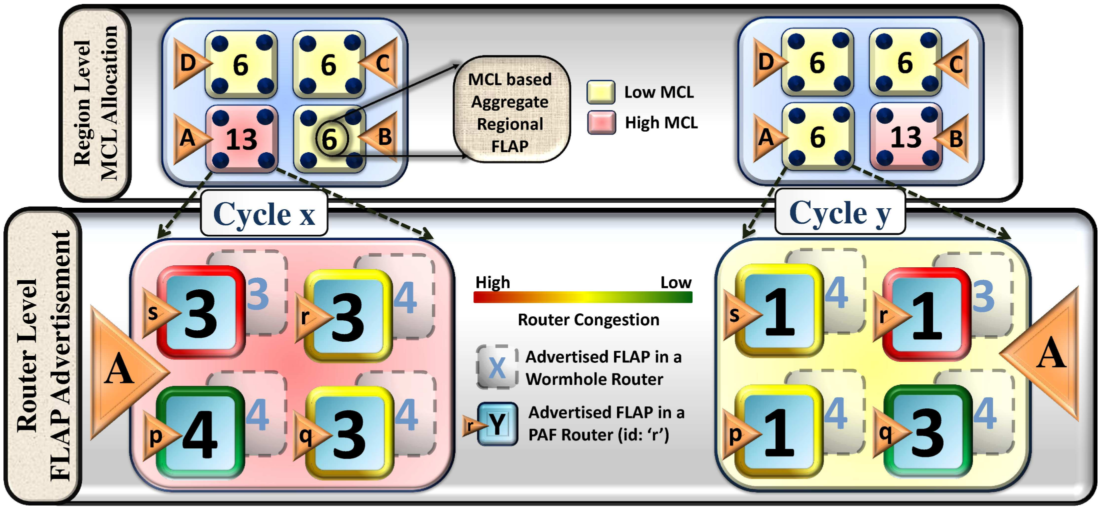

Region Level MCL Allocation
D
6
6
C
A
13
6
B
MCL based Aggregate Regional FLAP
Low MCL
High MCL
D
6
6
C
A
6
13
B
Cycle x
Cycle y
Router Level FLAP Advertisement
A
s
3
3
r
3
4
p
4
4
q
3
4
High
Low
Router Congestion
X
Advertised FLAP in a Wormhole Router
r
Y
Advertised FLAP in a PAF Router (id: 'r')
s
1
4
r
1
3
p
1
4
q
3
4
A

Fig. 3.5: Overview of PAF flow-control protocol. PAF allocates a high MCL to region A and low MCLs to the other regions in cycle x. In cycle y, only region B is allocated with a high MCL. At a router granularity, a high FLAP (4) is advertised by the least congested router p in cycle x. In cycle y, the router q is the least congested and advertises the highest FLAP (3). In both cycle x and cycle y, the aggregated FLAP of the routers corresponds to the respective MCL based regional FLAPs. The FLAP in Wormhole flow-control is congestion aware but agnostic of the regional load.

Hierarchical MCL Allocation

High concurrent switching of proximal regions is avoided by carefully adjusting the MCL allocated to each region. To realize MCL allocation principles at different granularities, a metric called ***Flit* *A*cceptance *P*otential (FLAP)** is defined. For a given input channel of a router, the $FLAP$ is set to 1 when it can receive an incoming flit (otherwise it is set to 0). For a router, the $FLAP$ indicates the aggregate $FLAP$ of its input channels. Similarly, the $FLAP$ of a particular region represents the aggregate $FLAP$ of the routers in that region.

At any given time, the $FLAP$ of a router employing wormhole flow control in a 2D mesh with four input channels is 4, when all of its input channels can receive at least one flit. PAF allocates variable MCL to each region by dynamically throttling their FLAPs, *irrespective of the space availability in the input channel's buffers.*

MCL allocation is a hierarchical process that can be applied at multiple spatial granularities. For example, a large region consists of many smaller sub-regions. The allocated MCL for the large region is distributed among the sub-regions, ensuring that proximal sub-regions are not simultaneously allocated with high MCLs. At the lowest granularity, each router's FLAP is managed in a manner that is consistent with the MCL allocation of the entire sub-region.

Illustrative Example

Figure 3.5 depicts the PAF technique using a 4X4 2D-mesh NoC, divided into 4 regions (A,B,C,D), each comprising 4 routers. In cycle x, PAF allocates a high MCL to region A and low MCLs to the proximal regions (B,C,D). To ensure a fair provisioning, PAF redistributes the MCL allocation in cycle y, so that region B is allocated with a high MCL, while its proximal regions are allocated with low MCLs.

The allocated MCL translates to a regional FLAP, which is distributed among the routers of a region. For example, in cycle x, a regional FLAP of 13 is distributed among the routers of region A. Router p advertises a FLAP of 4, while the other routers (q,r and s) advertise 3 FLAPs each.

Optimizations of PAF

The generic PAF technique needs multiple optimizations to efficiently tackle the design challenges (Section 3.3.1).

Minimizing Performance Impact: A few complementary approaches are explored that aim to retain a high performance in PAF.

- **Judicious FLAP Management:** To avoid a large flit delay in a given region, PAF allows intermittent high and low FLAPs in a router. For example, in contrast to cycle x, router q advertises more FLAP (3) in cycle y compared to the other routers.

- **Topological Awareness:** PAF can be adapted based on the network topology and expected traffic pattern. For example, central routers in a mesh typically experience a high resource demand. This demand can be met by allocating greater FLAPs to the central routers.

- **Congestion Awareness:** Two broad classification of PAF are explored.

Congestion Agnostic PAF: This variant of PAF statically allocates high and low FLAPs to the regional routers based on a round-robin fairness scheme. The FLAP allocation policy is not influenced by the network buffer occupancy. This variant is referred to as ***PAF-Static***.

Congestion Aware PAF: This variant of PAF manages the FLAP allocation based on the relative congestion of the network buffers. The following two congestion awareness are considered at different granularities.

Channel Granularity: The FLAP of the least congested channel of a router is set to 1, so that it can always receive an incoming flit. The other channels' FLAPs are dictated by the aggregate FLAP of the router. This variant of PAF is called ***PAF-CG***.

Router Granularity: The least congested router of a region is allocated with a high FLAP. However, the other routers are allocated with low FLAPs to avoid high simultaneous switching. The aggregate FLAPs of the routers is consistent with the allocated MCL of the region. For example, in cycle y in Figure 3.5, the least congested router q advertises more FLAP (3) compared to the other routers, each of which, advertises 1 FLAP. The aggregate FLAPs (6) of the routers match the allocated MCL based regional FLAP. This variant of PAF is referred to as ***PAF-RG***.

Avoiding Starvation: Repeated blocking of the flits at the same input channel of a router in successive cycles can cause a starvation. To avoid starvation, PAF adopts a round-robin fairness scheme to restrict flit reception across all the input channels of a router. Moreover, PAF uses deterministically routed escape VCs, allowing all the possible turns in the network without a deadlock situation.

Scalability: PAF is a hierarchical technique that uses local network information at the smallest regional granularity to ascertain the FLAPs of the routers. As the size of the smallest region remains the same even for a larger NoC, PAF can scale efficiently with the network size (Section 3.5.7).

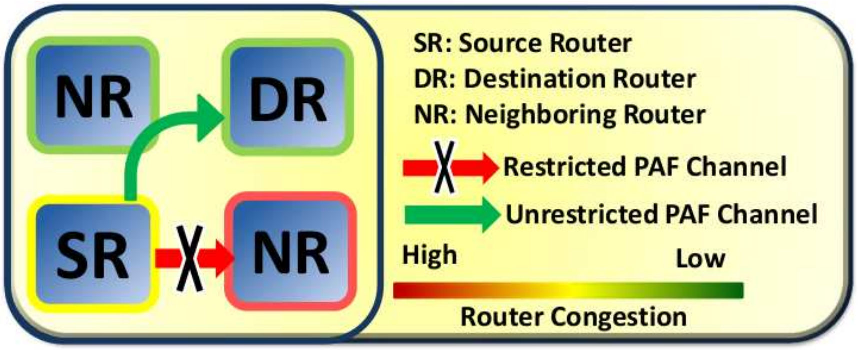


Fig. 3.6: PSN Aware Routing Algorithm (PAR).

3.3.3 PAF Aware Adaptive Routing Algorithm

Dynamically throttling the FLAP of a router may cause an intermittent upsurge in the local PSN due to an increased resource contention. A PAF cognizant routing algorithm, called PAR (*PSN-Aware Routing*), is proposed, to circumvent this upsurge, by steering the flit towards an unthrottled downstream path. Figure 3.6 depicts the conceptual overview of PAR. PAR primarily makes the routing decision based on the relative regional congestion information, aggregated solely along the minimal paths. If the chosen output channel has a throttled FLAP, PAR reroutes the flit to an orthogonal output channel, strictly maintaining the minimal path constraint. This strategy reduces local current spike and PSN by relieving router contention, but may occasionally increase the network latency by routing some flits towards more congested downstream paths. In a scenario, where both the minimal paths are blocked due to throttled FLAPs, the flit adheres to the initial channel assignment and waits in the upstream router for another cycle. PAR incurs no additional circuit overhead as it utilizes the same information required for PAF.

3.3.4 Advantages of PAF and PAR

In this section, the benefits from the proposed flow-control protocol (PAF) and adaptive routing algorithm (PAR) are summarized. PAF obviates the sudden congregation of flits in proximal regions of an NoC, by advertising high and low FLAPs in alternate epochs. As a result, events of large current surge are avoided, leading to a smooth noise profile of the

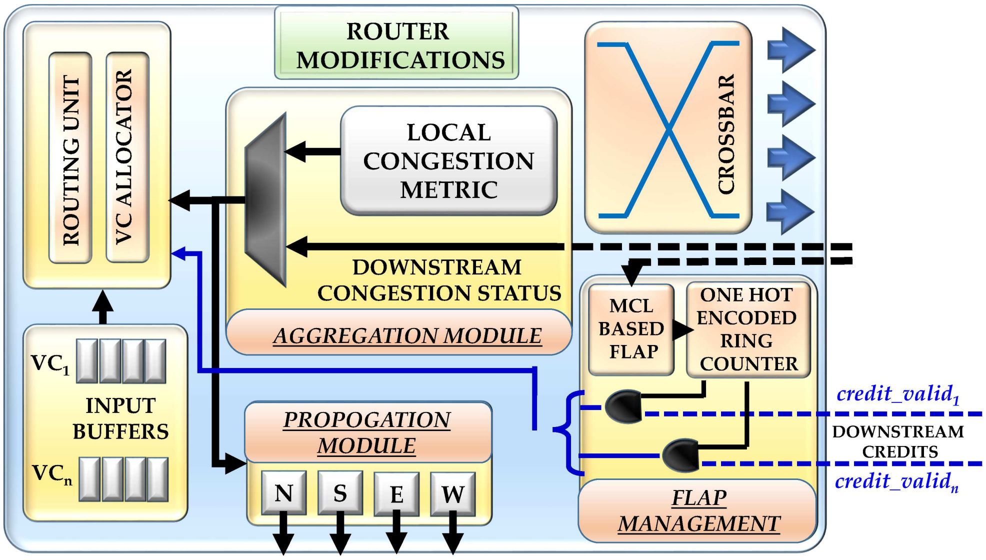


Fig. 3.7: IcoNoClast router micro-architecture.

NoC. However, PAF can unobtrusively aid to the local PSN in the upstream paths, due to a limited reception of flits. To further alleviate such local noise, PAR strives to route some flits towards the readily available downstream paths. A reduction in the peak noise essentially translates to a lower voltage guardband for fault-free operation of the NoC. As the routers will consume less energy under a lower voltage guardband, the proposed techniques promote energy efficient system design by moderating the maximum noise in the NoC.

3.3.5 Implementation

Figure 3.7 illustrates the implementation of an IcoNoClast router, that involves FLAP management and congestion management units.

- **FLAP Management:** Reception of flits in a router is managed by sending a *credit_valid* signal to the upstream router. A *credit_valid* signal, along with a statically managed, low overhead, round-robin logic is used to ascertain the FLAP of a router. Additionally, the *credit_valid* signal is fed with one of the output bits of a simple one-hot encoded ring counter, to sporadically restrict an incoming flit.

- **Congestion Management:** A low-bandwidth monitoring network is created to propagate the congestion information among the adjacent routers in a region. The monitoring network involves an *aggregation* and a *propagation* module at the router's low overhead port preselection logic [58]. The *aggregation* module combines the weighted congestion values from the downstream routers and the *propagation* module transmits the congestion information to the adjacent routers of a region.

3.4 Methodology

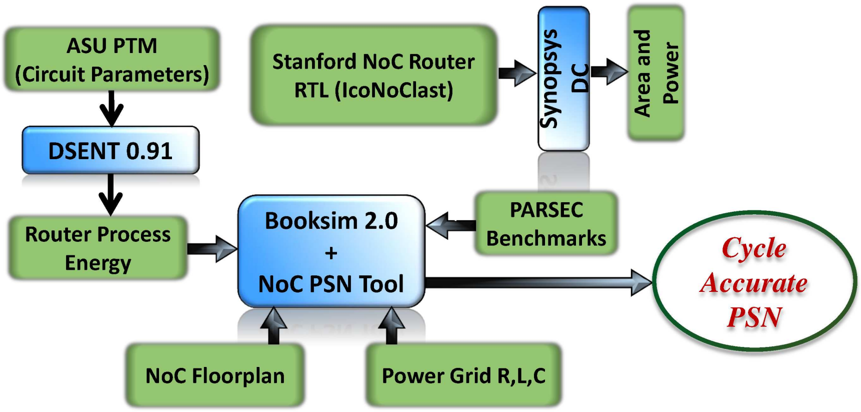


Fig. 3.8: Conceptual overview of the cross-layer methodology.

Figure 3.8 represents the hierarchy of the cross-layer methodology. The evaluation can be classified into two stages. Section 3.4.1 describes the PSN estimation technique and Section 3.4.2 discusses various performance metrics to evaluate the efficacy of the comparative schemes (Section 3.5.1).

3.4.1 Power Supply Noise Estimation

PDN Simulation: Challenges and Solution

SPICE based simulation of the PDNs of the modern VLSI circuits is computationally prohibitive, due to a large number of grid nodes and circuit elements. On the other hand, growing complexity of the PDNs (different topologies and on/off-chip capacitors, for example) makes it harder to accurately model them. Therefore, establishing a precise correlation between architectural events and voltage noise, is highly contingent on fast and accurate simulation of the PDN.

Many researchers in the past decade have proposed several models to curtail the simulation time while providing a reasonable accuracy for the node voltage. Using a fast and direct model proposed by Zheng et al. [59], Dahir et al. have recently devised a MATLAB based tool to estimate the PSN for the NoC power grid [10]. The power grid in this tool is modeled as a distributed RLC network. Constant voltage sources and switching capacitors are used to excite the network, and to model the on-chip activity, respectively. Zheng et al. have demonstrated that the model sped up the simulations by several order of magnitude compared to SPICE, with a maximum error of 5% in PSN calculation [59].

Cycle Accurate Noise Calculation

Dahir's PSN tool is integrated with Booksim2.0, to tightly couple the stages of architectural evaluation and PSN estimation. The consolidation of the two stages further boosts the simulation speed and helps diagnose the impact of architectural events on PSN. Cyclewise statistics of the voltage are generated at each power grid node for real workloads. The following data are collected for an accurate estimation of the PSN.

- ***Interconnect RLC Parameters:*** The R,L,C values of the grid interconnect are collected for the 32-nm technology node using the ASU PTM interconnect model [52]. The aspect ratio and pitch of the grid interconnect are obtained based on the ITRS interconnect predictions [57].

- ***Router Pipeline Energies:*** DSENT [51] is used to evaluate the energy of the router pipeline stages, using the router microarchitectural parameters for the 32-nm technology node. DSENT models an NoC router as a combination of input/output buffers, virtual channels, switch allocators and 2-stage crossbars.

- ***Traffic and Router Activity Dump:*** The codebase of Booksim 2.0 is instrumented, in order to dump various router activities (e.g., flit incoming, VC and switch allocation

Parameters	**Values**
Topology	8X8 regular 2D mesh
Virtual Channels per Port	8
Flit Buffers per VC	8
VC / Switch Allocator	iSlip
Traffic Workload	PARSEC Benchmarks
NoC Frequency	769 MHz

Table 3.2: Simulation parameters for performance evaluation.

etc.) at each cycle, by running PARSEC benchmarks on an 8X8 regular 2D mesh NoC. To mimic the traffic generated by multiple co-scheduled applications in an MPSoC, a heavy random traffic (with a flit injection rate of 0.15) is superimposed on top of the original application induced traffic of the PARSEC benchmarks.

3.4.2 Performance Evaluation

Table 3.2 details the simulation parameters used in the performance evaluation based on the following metrics.

Regional Peak PSN

When an application runs, each router in the network endures a varied peak PSN [10]. So, running the entire NoC with a single operating voltage to ensure 100% fault coverage is energy inefficient. On the other hand, providing each router with a separate operating voltage, increases the complexity and footprint of the voltage regulators. So, an 8X8 mesh NoC is divided into 16 regions, each containing 4 routers. Each region is assigned the minimum operating voltage, to ensure a fault-free communication. The regional peak PSN of the comparative schemes is evaluated in Section 3.5.3.

Average Network Latency

Booksim2.0 is used as the architectural simulator to run network simulations of the comparative schemes using real workloads. The simulation is run for 1 million cycles. The performance overheads of the comparative schemes are reported in terms of the overall average network latency in Section 3.5.4.

Energy Delay Product

Mitigating the peak supply noise reduces the minimum voltage guardband required for 100% fault coverage. As a result, all the routers in the network can operate at a reduced supply voltage and consume less energy. The improvement in the router energy is analyzed using DSENT [51]. Based on the network latency and the energy consumption of the routers, the energy efficiency is calculated using the Energy Delay Product (EDP) metric.

Area and Power

The RTL of the open source Stanford Verilog model [60] of a modern virtual channel NoC router is augmented to implement the IcoNoClast techniques. The router is assumed to be a part of a 2D mesh topology with 5-input/output ports and 8 VCs per port. The augmented router RTL is synthesized with the TSMC 45-nm library using Synopsys Design Compiler. From the synthesis report, the area and power overheads are calculated.

3.5 Experimental Results

In this section, the efficacy and overheads of various comparative schemes are analyzed (Section 3.5.1). First, the impact of a representative congestion aware routing scheme on the PSN is presented (Section 3.5.2). The improvement in regional peak PSN and performance overheads of IcoNoClast are discussed in Sections 3.5.3 and 3.5.4, respectively. The energy efficiency of the schemes is evaluated in Section 3.5.5. A comparison of the mean PSN for three synthetic traffic patterns at various injection rates is provided in Section 3.5.6. Finally, the area and power footprints of IcoNoClast are reported in Section 3.5.7.

3.5.1 Comparative Schemes

The comparative schemes are presented in Table 3.3. Each scheme is a combination of a flow-control protocol and a routing algorithm.

Schemes	Flow-Control	Routing Algorithm
Baseline	Wormhole	DBAR
PAF-SD	PAF-Static	DBAR
PAF-SP	PAF-Static	PAR
PAF-CD	PAF-CG	DBAR
PAF-CP	PAF-CG	PAR
PAF-RD	PAF-RG	DBAR
PAF-RP	PAF-RG	PAR

Table 3.3: Comparative schemes.

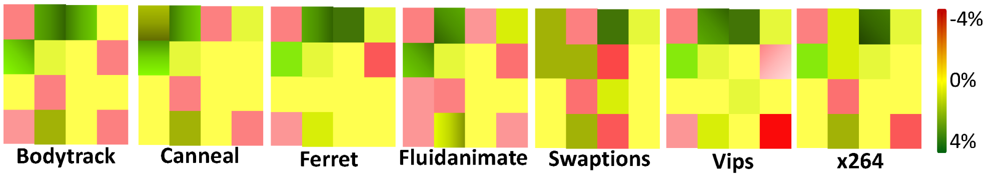


Fig. 3.9: Improvement in the regional peak PSN with DBAR compared to DOR. Green regions represent PSN improvement, while red regions represent PSN degradation.

3.5.2 Can Congestion Aware Routing Tackle PSN?

Figure 3.9 shows the improvement in the regional peak PSN with a representative congestion aware DBAR routing scheme compared to deterministic Dimension Order (DOR) XY routing. Both the routing schemes are used, along with wormhole flow-control. DBAR shows average peak PSN improvements of only 0.1-1%, across all the benchmarks. Some regions show worse peak PSN with DBAR, as DBAR cannot prevent intermittent influx of traffic to proximal uncongested regions, leading to an increase in the PSN.

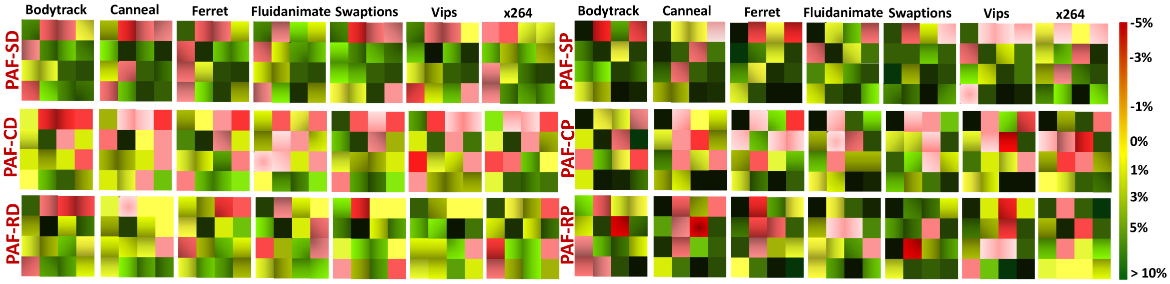


Fig. 3.10: Percentage improvement in regional peak PSN with respect to the baseline for comparative schemes. Each small square represents a region consisting of 4 routers. Reddish and greenish regions represent worse and improved peak PSN, respectively.

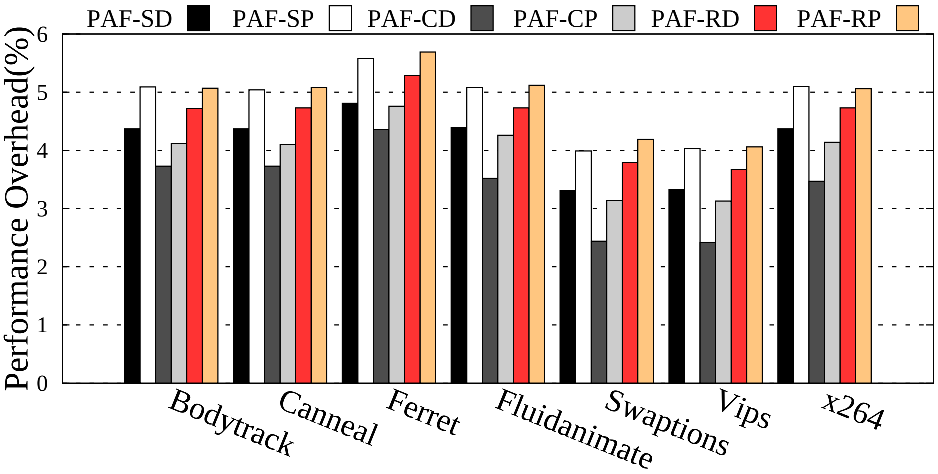


Fig. 3.11: Performance overhead (lower is better).

3.5.3 Regional Peak PSN Comparison

Figure 3.10 shows the percentage improvement in regional peak PSN of various comparative schemes, with respect to the baseline. The diversity of the improvement stems from a high degree of skew in the router loads for the real benchmarks. It can be noticed that PAF-SP, PAF-CP and PAF-RP show more pronounced improvements, as PAR can mitigate local PSN by reducing the intermittent upsurge in resource contention. In the PAF-SP scheme, considering the *ferret* benchmark, it is observed that 14 of the 16 regions see an improvement in peak PSN, and of these regions, 8 regions benefit from a peak PSN improvement greater than 10%. The reduction in peak PSN translates to a lower voltage guardband for these regions resulting in improved energy efficiency. The respective maximum regional PSN improvements observed in all the schemes (Section 3.5.1) are 8.1%, 13.2%, 6.6%, 14.8%, 7.8% and 14%, with respective average PSN improvements as 4.7%, 5.7%, 4.1%, 5.5%, 5% and 5.8%. Some regions show slightly worse peak PSN compared to the baseline, due to occasional increase in local congestion, incurred by the PAF.

3.5.4 Performance Overhead

Figure 3.11 shows the network latency overheads of the comparative schemes, with respect to the baseline. PAF-SP, PAF-CP and PAF-RP incur slightly more overheads,

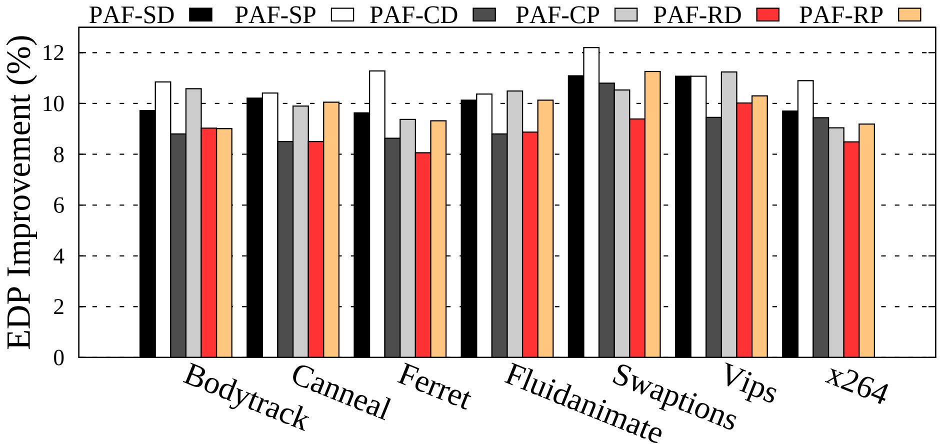


Fig. 3.12: EDP improvement (higher is better).

compared to the other schemes, as PAR sometimes takes more congested downstream paths in the network (Section 3.3.3). It can also be noticed that PAF-SD performs slightly better than PAF-RD due to PAF-Static's inherent fairness in allotting the FLAPs among the regional routers. As PAF-CG does not throttle the FLAPs of the least congested channels, both PAF-CD and PAF-CP incur very low performance overheads. There is a maximum performance degradation of 5.7% (*Ferret* in PAF-RP) with an average degradation of 4.1%, across all the schemes.

3.5.5 Energy Efficiency Comparison

Figure 3.12 shows the improvement in energy efficiency of the comparative schemes, in terms of EDP. To calculate the network energy, it is assumed that each region of the NoC is running with a minimum voltage guardband, required for fault free communication (Section 3.4.2). The guardbands are dictated by the peak PSN observed with the pertinent schemes. It can be noticed that all the variants of PAF incur better EDP, when used along with PAR routing (PAF-SP, PAF-CP and PAF-RP). A maximum EDP improvement of 12.2% (*Swaptions* in PAF-SP) is observed, with an average improvement of ∼10%, across all the schemes. PAF-SP shows maximum improvements in EDP, among all the schemes.

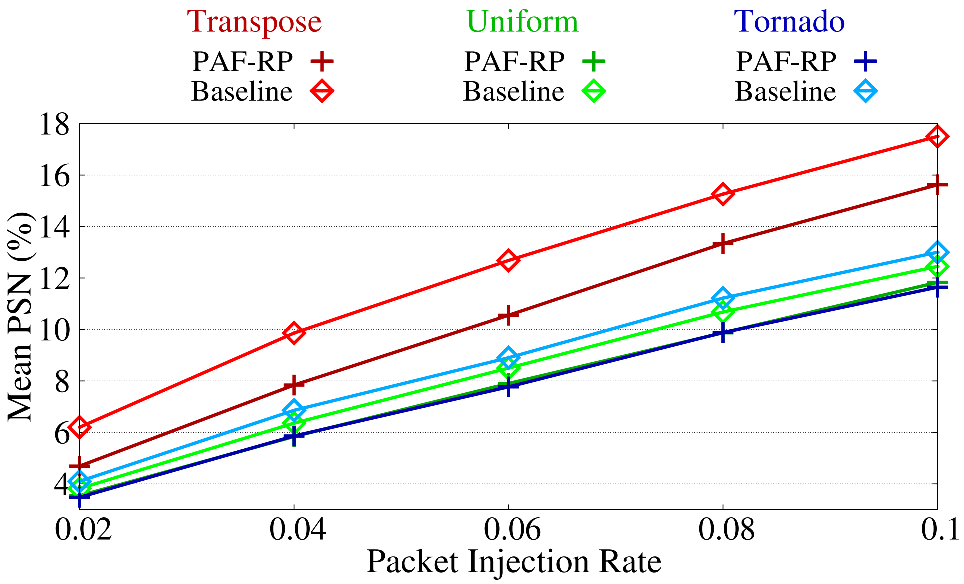


Fig. 3.13: Mean PSN for three traffic patterns.

PAF variants	Area	Power
PAF-Static	0.10%	0.16%
PAF-CG/PAF-RG	1.42%	2.38%

Table 3.4: PAF overheads from the synthesized hardware.

3.5.6 Mean PSN Comparison for Synthetic Traffics

Figure 3.13 demonstrates the variation of the mean PSN with packet injection rate, for three traffic patterns (*Transpose*, *Uniform* and *Tornado*), employing baseline and PAF-RP schemes. The number of flits per packet is 20. In general, the PSN increases with the injection rate due to increased switching activity in the routers at higher injection rates. It is observed that PAF-RP consistently incurs lower PSN compared to the baseline, at all the injection rates. The reduction in PSN (with PAF-RP) also varies across the traffic patterns, and the most PSN mitigation at the highest injection rate is observed in *Transpose*.

3.5.7 Area and Power Footprint

Marginal overheads are reported from the synthesized hardware of the PAF variants (Table 3.4). The congestion management unit incurs more overhead, compared to the simple FLAP management unit.

CHAPTER 4
FOSTERING ENERGY EFFICIENCY IN A NEAR-THRESHOLD GPU THROUGH A TACTICAL PERFORMANCE BOOST

4.1 Background and Contributions of This Work

GPUs have demonstrated substantial performance advantages over CPUs, by exploiting large *thread-level parallelism* in data-intensive, highly parallel applications. With this immense performance advantage, a GPU's power consumption has also grown steadily, reaching 300W [61]. On the other hand, recent advances in *Near-Threshold Computing* (NTC), where the supply voltage is set slightly above the device threshold voltage, have shown a great promise in radically curtailing the chip power consumption. This fantastic improvement in energy efficiency of NTC circuits does come with a steep performance loss. To compensate for the performance loss in a single device, NTC circuits generally employ more devices to execute in parallel [3]. Consequently, a GPU is of particular interest at NTC, as it is built to exploit parallelism.

While conceptually intriguing, a GPU design for NTC presents two fundamental challenges. First, elongated delays in NTC circuits make the GPU applications severely sensitive to Multi-cycle Latency Datapaths (MLDs) within the GPU pipeline. When a GPU thread heavily utilizes one of these MLDs, like the functional units for transcendental operations, for example [62], the entire application performance can become latency sensitive, obliterating the advantages of parallel execution. Second, *Process Variation* (PV) presents a tremendous design challenge for NTC systems. In fact, a wider spatial-spread of the cores of a GPU can further exacerbate the core-wise performance asymmetry from PV at NTC.

To tackle these challenges, this chapter identifies the unique opportunities arising at the near-threshold regime. For example, the emerging NTC era ushers an intriguing possibility of up to 10X improvement in circuit performance, at a moderate energy cost [5].

Such a large performance enhancement was never a possibility at STC, since STC circuits already operate near their minimal delay region [5]. By systematically exploiting such device level characteristics at the circuit-architecture layer, this chapter presents *SwiftGPU: an energy-efficient GPU design paradigm for NTC.* Inherently PV-aware, SwiftGPU tackles key challenges of NTC GPU designs, by dynamically speeding up the MLDs, and manipulating the thread level parallelization. Collectively, these techniques mitigate the performance sensitivity to MLDs, and performance imbalance from PV, substantially improving energy efficiency at NTC.

The specific contributions in this chapter are discussed next.

- Emerging performance and energy-efficiency hazards are discussed as GPUs operate at NTC (Section 4.2).

- SwiftGPU, a collection of low-overhead design time and run-time solutions, is proposed, to address the emerging challenges in an NTC GPU. By detecting the utilization pattern of MLDs in a stream core, SwiftGPU selectively speeds up their execution in the stream cores to improve the NTC GPU energy efficiency (Section 4.3).

- Using an elaborate cross-layer methodology (Section 4.4), it is demonstrated that an average improvement of 14.8% in energy efficiency can be achieved, over an ideal PV-free STC GPU, across a range of emerging general purpose GPU (GPGPU) applications. Using synthesis, place and route of a GPU RTL, augmented with SwiftGPU, the area, wire-length and power overheads are reported to be 0.65%, 2.6% and 3.7%, respectively (Section 4.5).

4.2 Motivation

In this section, the emerging efficiency hazards for NTC GPUs are explored (Section 4.2.1). Using a rigorous cross-layer methodology (Section 4.2.2), the GPU performance trends are analyzed at NTC (Section 4.2.3). The potency of a strategic performance boost is shown to improve the energy-efficiency of NTC GPUs (Section 4.2.4).

4.2.1 Efficiency Hazards in NTC GPUs

An increase in the number of Compute Units (CUs) [1], is expected to recoup the performance loss due to the frequency degradation at NTC. A key research question is, *whether the emerging GPGPU applications can exploit the increased thread level parallelism at NTC, to sustain the STC efficiency.* A few performance bottlenecks are discussed that plague energy-efficient operation of NTC GPUs.

- **Nature of Datapath Usage:** Compute-intensive GPGPU applications are likely to exhibit a high MLD utilization and increased sensitivity to their latencies. Consequently, operating a GPU at NTC will prove to be an inefficient design choice, unless the execution of the MLDs can be selectively sped up.
- **Thread-level Data Dependency:** As an NTC GPU has more cores than its STC counterpart, each core in the former will have less concurrent threads to execute, potentially escalating the data-dependency within a CU. The increased data-dependencies can make the execution *latency sensitive*, degrading the application performance.
- **Process Variation**: NTC circuits are susceptible to within-die process variation, severely altering the CU frequencies and leakage power of the GPU. According to Chang et al. , the maximum within-die frequency variation at 22-nm, can be ∼200%, while operating at the NTC region [63]. It is imperative to efficiently counteract the effect of PV in an NTC GPU, to sustain a high performance per watt.

4.2.2 Methodology

Multi2Sim 4.2 [64] is used to model AMD's Evergreen architecture GPU – Radeon 5870. The GPGPU applications from AMD's Accelerated Parallel Processing SDK suite [65] are used for performance evaluation. Based on the frequency scaling factors in previous works [66, 67], the CU frequency at NTC is set to be 25% of the CU frequency at STC.

[1] A *Compute Unit* is the fundamental unit of computation, comprising multiple SIMD units, referred to as *stream cores*.

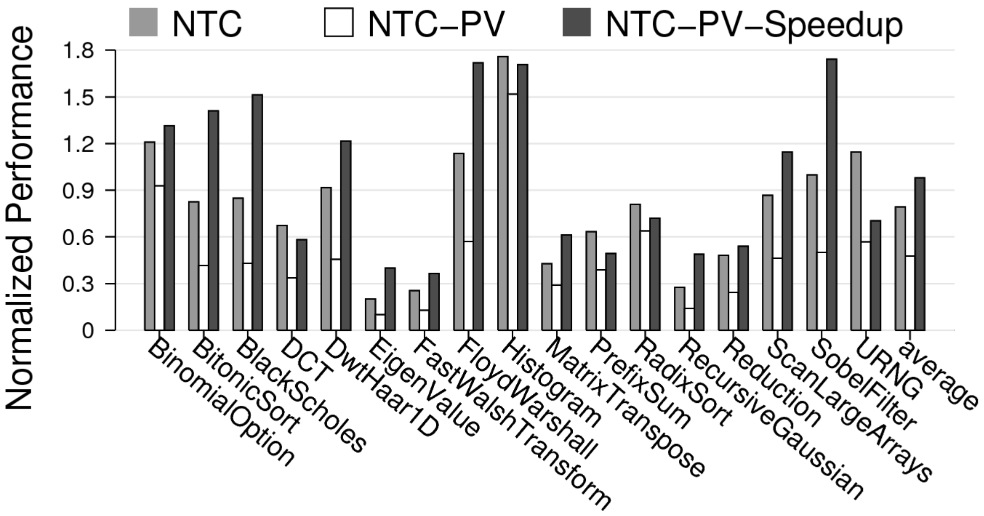


Fig. 4.1: Performance of NTC GPU (Normalized to STC GPU).

To maintain *identical theoretical compute bandwidth across the NTC and STC systems*, the number of CUs at NTC is set to be 4X as that of the STC. The original MLD latencies are between 4 - 40 cycles, for both STC and NTC. To model process-variation, VARIUS-NTV [68] is used. A detailed methodology description is discussed in Section 4.4.

4.2.3 NTC Performance Trend

Figure 4.1 shows the performance results of three cases: NTC (*PV-free NTC GPU*), NTC-PV (*PV-infected NTC GPU*) and NTC-PV-Speedup (*PV-infected NTC GPU with 4X MLD boost*), normalized to a baseline *PV-free STC GPU*. All the benchmarks exhibit a worse performance in the PV-infected NTC GPU, compared to the baseline. A maximum performance degradation of 90% is observed in *EigenValue*. It can also be noticed that the solo impact of PV, brings about severe performance loss with respect to the PV-free NTC GPU. However, 8 out of 17 benchmarks, perform better than the baseline, when all the MLDs are statically boosted by a factor of 4X. For example, *SobelFilter* exhibits a staggering 80% performance improvement with the MLD speedup. The performance degradations in *EigenValue* and *FastWalshTransform*, are also reduced from about 90% to about 60%, by the MLD speedup.

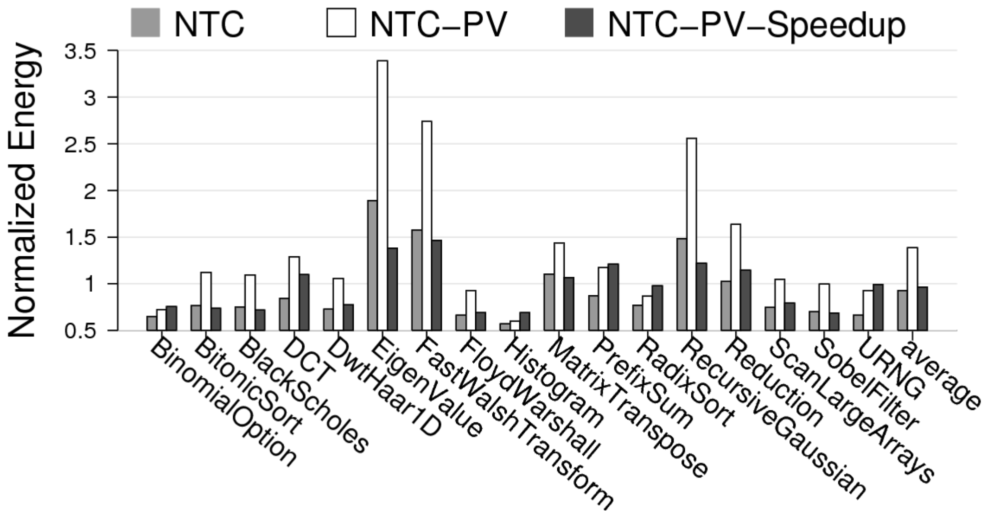


Fig. 4.2: Energy of an NTC GPU (Normalized to an STC GPU).

4.2.4 Significance

The initial results reveal that the emerging GPGPU applications are very sensitive to various NTC hazards, as well as, to the variations in the MLD latencies (Section 4.2.3). Such a performance sensitivity can have a profound impact on the energy efficiency of the GPUs. Figure 4.2 shows a preliminary investigation of this impact. With a static 4X boost applied to the MLDs, the PV-infected NTC GPU has a 31.5% reduction in the average energy consumption, which is 3.5% lower than the energy consumed by the PV-free STC GPU. This energy benefit comes from a substantial reduction in the leakage energy, a crucial manifestation of the MLD boost. Inspired by these circuit-architectural insights, this chapter explores SwiftGPU—a novel energy-efficient GPU design paradigm at NTC. SwiftGPU incorporates *Self-Adaptive Sprint (SAS)* technique, to selectively speed up the processing in the stream cores, to improve the NTC GPU efficiency.

4.3 Self-Adaptive Sprint in NTC GPUs

This section discusses Self-Adaptive Sprint (SAS). SAS promotes an energy-efficient execution, by tackling the NTC hazards in GPUs. The overview of SAS is presented in Section 4.3.1 and its circuit-architectural aspects are explained in Sections 4.3.2 and 4.3.3. The dynamic control strategies of SAS are outlined in Section 4.3.4.

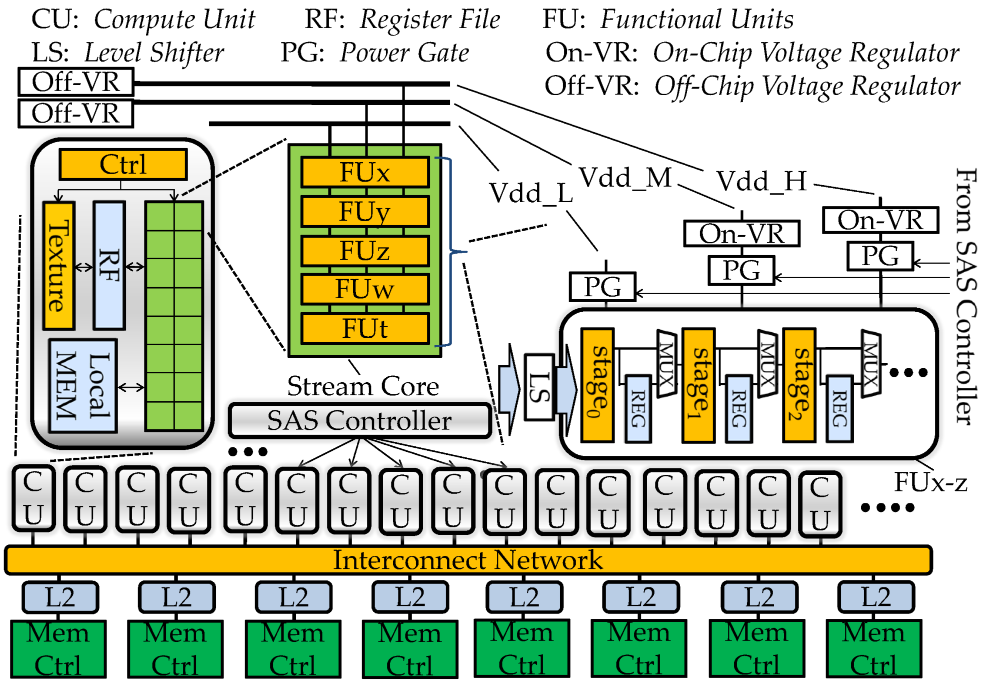


Fig. 4.3: Overview of SAS.

4.3.1 Overview

Figure 4.3 gives an overview of the proposed SAS technique. The *SAS Controller* dynamically manages the execution speed of the CU MLDs. To tackle the impact of PV, a number of crucial design strategies are adopted, ranging from the use of tunable voltage rails, to a meticulous selection of the MLD speeds. To support several datapath speeds, the underlying power-delivery network is augmented to allow three different supply voltage rails: *Vdd_H*, *Vdd_M* and *Vdd_L*, respectively. The SAS controller monitors the runtime hardware utilization of various CU MLDs, and dynamically adjusts the MLD speed to improve the energy efficiency of the entire system. A few key design aspects of SAS are outlined next.

4.3.2 Circuit-level Support for SAS

Providing multiple voltage-rails in the CU datapaths raises several critical design challenges, discussed next.

- The specific voltages offered in the rails need to be selected based on the technology node, circuit level delay characteristics of underlying devices, and associated overheads. For example, for the 22-nm technology node and a nominal frequency of 175 MHz, it is determined that *Vdd_H*, *Vdd_M* and *Vdd_L* can be set to 0.6V, 0.42V and 0.35V, respectively, to enable 4X, 2X and nominal (1X) sprint speeds in the CU MLDs. Three off-chip voltage regulators are employed for the nominal and higher voltage rails [69]. Using a transition-time test setup similar to [69], it is observed that the switching between different supply voltages can complete within one cycle (5.7 ns) of the NTC GPU.

- The voltage required to sustain a specific sprint, is likely to encounter a spatial variance due to the PV, compounding the complexity of online boost. To address this problem, *Vdd_H*, *Vdd_M* and *Vdd_L* are ascertained and recorded, for all the CUs, after fabrication. Each CU is endowed with its own low-overhead on-chip voltage regulator [70], to control only its MLD sprint. As the three off-chip regulators can deliver the required boost at the nominal frequency, such on-chip regulators can essentially work at a narrow voltage range, preserving a high conversion efficiency. Moreover, the energy overhead associated with the voltage conversion, is significantly reduced due to (a) relatively low power consumption of the MLDs (compared to the entire CU), and (b) sporadic occurrence of the MLD boost. During the kernel execution, the SAS controller boosts the MLDs, by selecting the appropriate voltage for a CU.

- It is imperative to carefully consider the transition time between different voltage rails. To accomplish runtime transitions, the voltage rails are augmented with a set of low-overhead level-shifters, connected to the CU MLD components. Such a level-shifter consists of 24 transistors, and only adds a marginal delay to the original MLDs [71].

4.3.3 Micro-architecture Support for Functional Correctness

Allowing variable latencies in the datapaths creates several intriguing design challenges in the micro-architecture, in order to retain its functional correctness.

- A multi-cycle datapath may complete its computation within a single cycle under a sprint mode. However, many of the existing CU MLDs are pipelined, thereby preventing the high-speed computation to propagate to the output. To resolve this issue, SAS dynamically allows intermediate pipeline registers in these datapaths to become transparent, forwarding the high-speed computation to the output.

- SAS must allow issuing dependent instructions in a timely manner under the sprint mode, so as to avoid unnecessary delay. Instruction execution is often tracked using a fixed set of countdown registers. These countdown registers are altered carefully, in order to reflect speedy execution and subsequent issue of dependent instructions in the pipeline.

- During the transition between different voltage rails, the pipeline operations may become unstable. To prevent incorrect execution, SAS initiates a pipeline flush before a transition. The overheads from these pipeline flush operations directly influence the SAS control strategy (e.g., high overhead prohibits frequent transitions).

4.3.4 Dynamic Control of SAS

A key challenge in the design of SAS is dynamically deciding the right boost for speeding the datapaths in a CU. A few aspects of this intriguing research challenge are outlined next.

Necessity of Dynamic Control

The utilization of MLDs in the GPU processors can vary substantially, altering the effect of their boost on the GPU performance. A GPU application consists of one or more kernels. A spatio-temporal variation is observed in the core pipeline utilization, during a given kernel execution. Three major elements of these variations are: (a) spatial variation of a given MLD utilization across different *thread groups* [2]; (b) temporal variation of a given MLD within a single thread; and (c) different utilization of MLDs across thread groups. Moreover, process variation gives rise to a large difference among the maximum operating

[2] A *thread group* is defined as the collection of threads belonging to the same instruction instance. It is equivalent to *warp* according to NVIDIA and *wavefront* according to AMD.

frequencies of the CU cores. To exploit the widely varying performance gains from MLD speedups, and the PV induced performance variation, a dynamic control of SAS is explored.

Estimating Maximum Boost

The initial investigation has revealed that a static boost of 4X can effectively tolerate the PV, while delivering a comparable or better performance compared to a PV-free STC GPU (Section 4.2.3). It is also noticed that the performance sensitivity to MLD boost, steeply declines at the higher sprint speeds. Moreover, over-boosting the MLDs, can potentially aggravate the overall performance per watt, due to its high energy overhead. Therefore, in this work, the maximum sprint speed is limited to 4-times the nominal sprint.

Predicting Boost

The kernel characteristics, the MLD utilization pattern and PV induced performance heterogeneity are used to estimate the amount of boost.

Kernel Characteristics: A preliminary investigation (Section 4.2) reveals that the performance sensitivity of a GPU to an MLD, depends on the number of concurrent threads executing on the stream core. A small number of concurrent threads, is unlikely to amortize the MLD latency due to a greater thread-level data-dependency across the CUs (Section 4.2.1). Given the block size *(b)* and the volume of the concurrent blocks *(V)* of the kernel, as well as, the volume of the stream cores on the GPU (V_c), the number of concurrent threads on a stream core (t_C) can be expressed as: $t_C = b\frac{V}{V_c}$.

A boost in a CU MLD is expected to improve the performance of the GPU only if, t_C is lower than the MLD latency. When a kernel is launched, its dimensions are identified (usually specified in the kernel), and set the maximum sprint speed (*maxSprintSpeed*) of an MLD, according to Algorithm 1, where *modalFULatency* is the modal latency of the FUs (4 cycles in this work).

MLD Utilization: Due to variation in the utilization of the MLDs, SAS must be able to control the sprint speed of each MLD independently. To predict the sprint speed of each MLD, it tracks the usage of each MLD during every interval (100,000 cycles). At the end

Algorithm 1 Set Maximum Sprint Speed

```
if $t_C$ <modalFULatency/2 then
    maxSprintSpeed = 4X
else
    maxSprintSpeed = 2X
end if
```

of each interval, the usage of each MLD is compared to a threshold (a certain percentage of the *instruction bundles*[3]). The MLDs with a higher usage than the threshold value for a particular interval, are given a sprint boost in the next interval.

Spatial Performance Heterogeneity: The performance variance of the CU cores in SAS can be exploited to mitigate the impact of PV. For example, the slowest CUs are power-gated, if they are not utilized by the running application. A two-pronged strategy is adopted for the remaining CUs, executing a kernel. First, the sprint of the slowest CUs is controlled by their dynamic MLD utilization patterns. Second, to ensure a spatially balanced performance, the sprints of the faster CUs are optimized, based on their respective frequency differences from the slowest CU.

Improving the Energy Efficiency of SAS

The energy overhead associated with a higher sprint boost, can potentially reduce the effective performance per watt. In order to sustain an energy-efficient execution, a low-overhead monitoring network is created, that periodically adjusts the sprint speed, by approximately assessing the energy-efficiency of the last epoch. If a new sprint-speed offers a better efficiency (reflected by a certain percentage improvement), SAS maintains the new sprint-speed till the next check. A degradation in efficiency leads to a switch-over to the older sprint-speed for an extended period of time. The iteration interval and the sprint-speed are optimized, based on specific kernel behavior.

For a decrease in sprint speed, the MLD latency slots in each stream core are doubled. At the same time, SAS power-gates half of the active stream cores within a CU, and migrates their threads to the remaining cores [17]. In this way, each thread group will be folded and

[3] An *instruction bundle* is a set of simultaneously issued instructions in a CU.

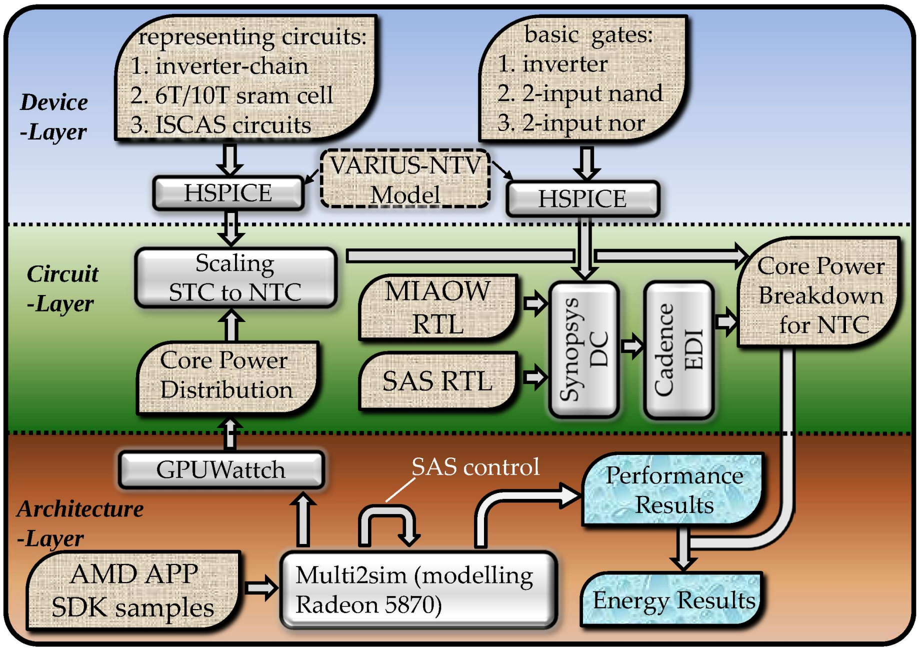


Fig. 4.4: Overview of the cross-layer methodology.

executed on the remaining cores in a round-robin fashion. On the other hand, an increase in sprint speed causes the threads to sprawl onto more stream cores. Upon thread migration, the dirty register data on a stream core will be written back to the local memory. This write-back can take up-to 1024 cycles, depending on the size of the dirty register data. This overhead is considered while evaluating the performance of SAS (Section 4.4).

4.4 Methodology

Figure 4.4 shows the overview of the methodology. Multiple layers (e.g., architecture, device and circuit layer) are considered to rigorously evaluate the efficacy of the proposed techniques. Specific components of each layer are briefly outlined next.

Parameter	**STC**	**NTC**
Frequency	700MHz	175MHz
# of stream cores	320	1280
Thread Group Size	64	64/32/16
MLD latency	4- 40 cycles	4- 40 / 1- 10 cycles
Local memory	32Kb, latency: 2-cycles	32Kb, latency: 2-cycles
L2 cache	8X256Kb, latency: 20 ns	8X256Kb, latency: 20 ns
Device Memory	B/W: 25.6GB/s, latency: 100 ns	B/W: 25.6GB/s, latency: 100 ns

Table 4.1: GPU configuration.

4.4.1 Architecture Layer

Multi2Sim is used as the architectural simulator [64] . Depending on the kernel dimensions, as well as, the FU usage in each interval, the MLD latency is dynamically changed (Section 4.3.4). For each benchmark, the GPU kernel is run for 1 iteration, considering all the possible MLD speedups. The statistics from Multi2Sim is used as inputs to GPUWattch [72], to evaluate the power consumption of the GPU components. The performance and energy consumption of each interval are used to calculate the overall energy efficiency of SAS. Table 4.1 lists the architectural parameters for the STC and NTC GPUs, respectively. The thread group size at STC is statically set to 64. At NTC, the thread group size is adjusted for different benchmarks, to exploit the increased volume of the stream cores. From STC to NTC, the volume of the stream cores is increased, and their frequency is decreased, by a factor of 4X. With this configuration, the memory traffic is expected to be similar in the STC and NTC GPUs. Therefore, identical configurations for the memory components, outside of the stream core (i.e., L2 cache and Device Memory), are used, for both STC and NTC GPUs.

4.4.2 Device Layer

VARIUS-NTV [68] is used to model PV-induced CU-wise performance variation. Based on a cumulative distribution of the CU speed for 2,000 NTC GPU instances, a collection of 80 CUs are randomly generated, to represent a PV-affected NTC GPU. The fastest CU in the NTC GPU can run at 3X higher speed than the slowest CU. To evaluate the NTC energy consumption, HSPICE simulations are performed on various basic gates and circuits, for the 22-nm technology node [73]. The simulation considers (a) a canonical 31-stage FanOut-of-4 (FO4) inverter-chain, and ISCAS85 circuits, to represent various combinational logic in GPUs; (b) a 6T-SRAM cell and a 10T-SRAM cell [74, 75] to represent the memory configurations at STC and NTC GPUs, respectively. Previous works have shown that interconnect power is ∼50% of the core dynamic power at STC [76]. This percentage is assumed to remain the same at NTC, as voltage scaling equally affects the interconnect and the core dynamic power.

4.4.3 Circuit Layer

The power results of GPUWattch and the results from the gate-layer simulations are combined to estimate the power consumed by an NTC GPU. The power values of each GPU component are then scaled into NTC, according to the power scaling trend obtained for the representing circuits (Section 4.4.2). To evaluate the overheads, a reference GPU RTL [77] is augmented with SAS. Place and route phases are performed with Cadence SoC Encounter, to get a more accurate estimation of the hardware overheads of SAS.

4.5 Experimental Results

In this section, the efficacy and overheads of various comparative schemes are analyzed (Section 4.5.1). Section 4.5.2 presents the empirical evaluation of the usage threshold for SAS. Section 4.5.3 and 4.5.4 discuss the performance and energy-efficiency of various schemes, respectively. Finally, Section 4.5.5 presents the hardware implementation overheads of SwiftGPU.

4.5.1 Comparative Schemes

- **Frequency Screening (FS)**: This scheme loosely models the technique proposed in [78]. The GPU is run at a higher frequency, by disabling the slowest stream cores.

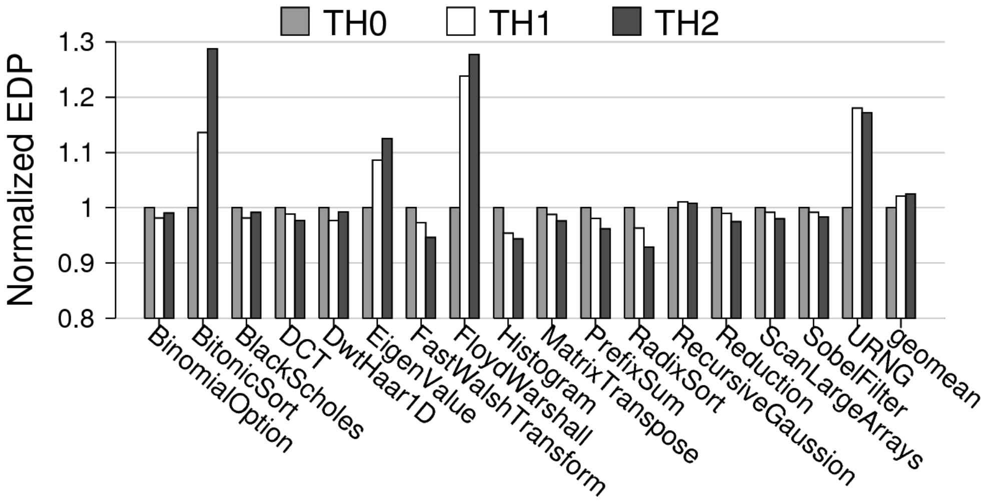


Fig. 4.5: Energy-efficiency of different usage thresholds. Lower is better. Three thresholds are tested: TH0(20%), TH1(30%), TH2(40%). Results are normalized to TH0.

- **Thread Gather (TG)**: This scheme resembles the core power-gating technique proposed in [17]. The threads are squeezed to a lesser number of CUs, to reduce the idle time in each CU (Section 4.1). The idle CUs are power-gated.

- **Static Sprint Execution (SSE)**: SSE employs a static single sprint mode for each benchmark. The sprint mode is set based on the dimensions of the executing kernel.

- **SAS**: This is the proposed technique in this chapter. It incorporates TG and dynamic sprint control, described in Section 4.3.4.

4.5.2 Selecting Threshold For Dynamic Sprint

A single usage threshold is empirically selected for all the FUs in the dynamic sprint control (Section 4.3.4). Three different thresholds, 20%, 30%, 40%, are explored and their respective energy-efficiencies in terms of the Energy Delay Product (EDP) are measured. The sprint speed is ascertained by the dimensions of the executing kernel. Figure 4.5 shows that, all the thresholds have marginal differences in energy efficiency (less than 5%), in 10 out of 17 benchmarks. However, a threshold of 20% offers remarkably better energy-efficiencies in a few benchmarks (e.g., *BitonicSort*). So, a 20% usage threshold is chosen in the dynamic sprint control.

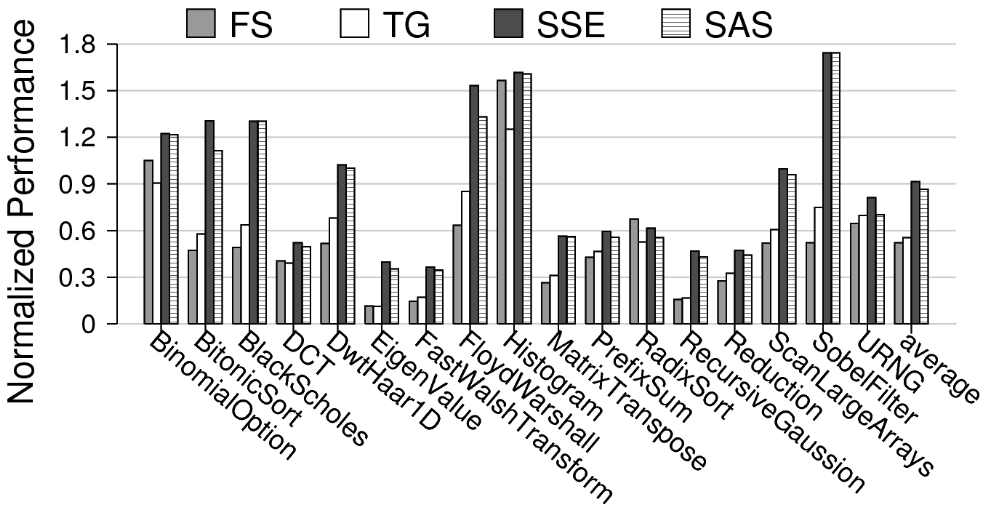


Fig. 4.6: Performance Comparison. Higher is better.

4.5.3 Performance Results

Figure 4.6 depicts the performance of various comparative schemes (Section 4.5.1). The results are normalized to an ideal PV-free STC GPU, without any enhancements. Both FS and TG deliver remarkably degraded performance than the proposed techniques (SSE and SAS), as they employ less number of cores, and do not perform sprinted MLD execution. Although both FS and TG offer an overall similar performance, they extensively diverge for several benchmarks. SSE globally outperforms all the other schemes. The performance of SAS is lower than that of the SSE for most of the benchmarks, as sprint execution is dynamically throttled in SAS. *Despite a 75% reduction in the operating frequency, the average performance of SAS enabled NTC-GPU is only 13.4% lower than that of the baseline PV-free STC GPU, across all the benchmarks.*

4.5.4 Energy-Efficiency Results

Figure 4.7 shows the energy benefits of all the comparative schemes, compared to the baseline PV-free STC GPU. For most benchmarks, both FS and TG consume more energy than the proposed techniques. A major source of this energy over-consumption comes from leakage, which is a significant fraction of the total energy in NTC circuits [3, 21], and proportional to the application execution time. However, in several benchmarks (e.g.,

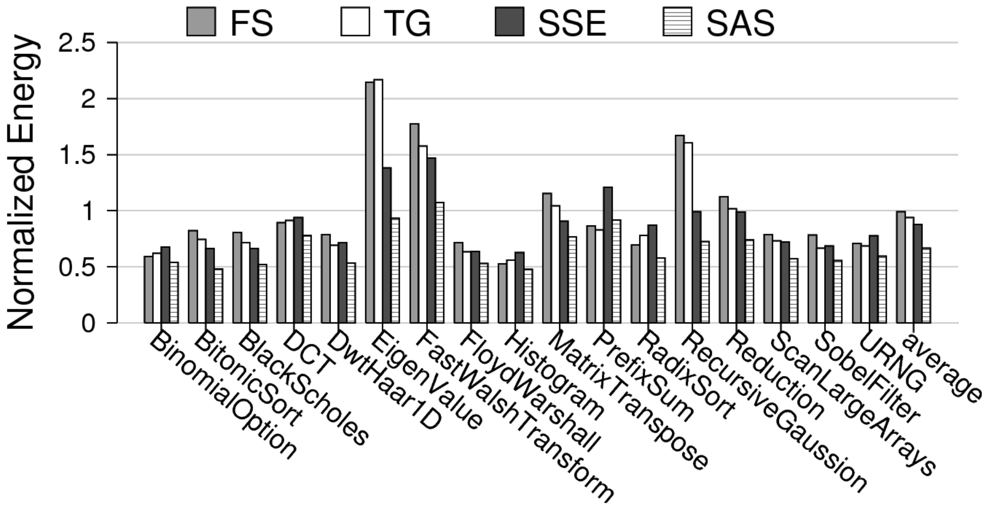


Fig. 4.7: Energy Comparison. Lower is better.

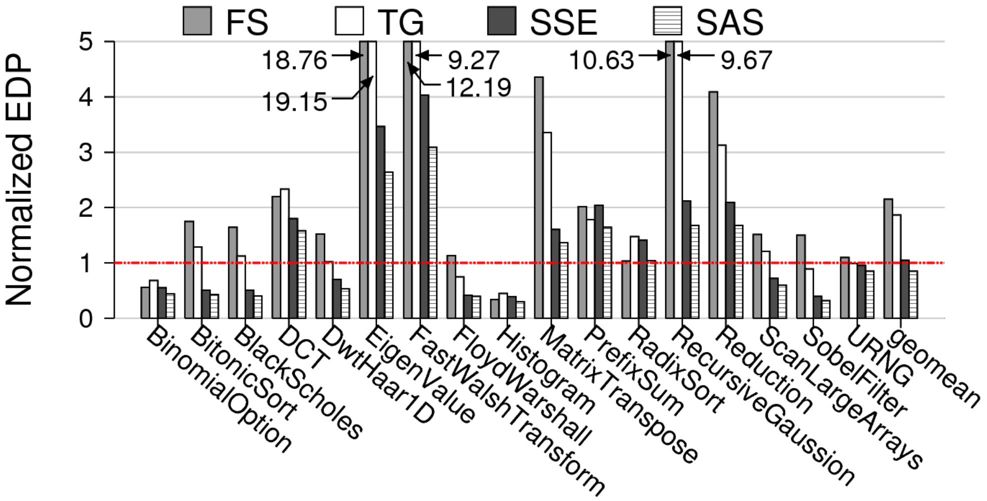


Fig. 4.8: Energy-efficiency Comparison. Lower is better.

PrefixSum), the proposed techniques consume more energy than FS or TG. Such results signify the inefficacy of employing sprint execution in the slow cores. Therefore, it is prudent to power-gate the slower cores, and execute all the threads in the faster cores, at a higher frequency. Across all the benchmarks, SSE always has the higher energy consumption than SAS, as the former always enables sprint execution in all the MLDs. Overall, the average energy consumption of SAS is 33.6% lower than the baseline STC GPU.

Figures 4.8 illustrates the energy efficiency of the comparative schemes, in terms of EDP. The results are again normalized to the baseline STC GPU. The escalated performance and the reduced energy consumptions of the proposed schemes, compound to an overwhelming benefit in EDP, compared to FS and TG. SAS shows an 18.9% improvement in EDP over SSE, as SSE is more power-hungry than SAS. *SAS also outperforms the baseline STC GPU, with an EDP improvement of 14.8%.*

4.5.5 Implementation Costs

The hardware cost of SAS can be attributed to two major factors: the sprint execution infrastructure (Sections 4.3.2 and 4.3.3) and the SAS control framework (Section 4.3.4). As usually available in modern GPUs, the implementation cost of pipeline flush and instruction replay, as well as, the performance counters, that track the FU usage, are excluded. Evaluated with the cross-layer methodology, the area, wire-length and power overheads for SAS implementation are 0.65%, 2.6% and 3.7%, respectively, compared to the original GPU.

CHAPTER 5

UNCOVERING THE PARADIGM SHIFT IN SECURITY VULNERABILITY AT NEAR-THRESHOLD COMPUTING

5.1 Background and Contributions of This Work

Evolution in low-power computing over the last decade has resulted in an unprecedented integration of electronic devices with human beings, fostering a sustainable digital environment. Owing to the benefits of Dennard scaling and a rapid advancement in packaging technology, the researchers foresee a ubiquitous adoption of Body Area Networks, comprising low-power, high-performance computing devices, such as, Internet of Things (IoT) [79]. This tremendous revolution in human-computer proximity, however, raises serious security concerns, stemming from the low-power operations of the silicon devices. In several critical applications (e.g., implantable medical devices), a single security breach can be detrimental. Although, the research in energy-efficient computing has matured over the years, understanding the security vulnerabilities of low-power devices is still at a nascent stage. In this work, real hardware-based experiments are performed that demonstrate an increasing security vulnerability as one moves to low-power systems.

Near-Threshold Computing (NTC)—a promising platform for low-power systems—has recently gained traction in the research community [3]. However, moving from traditional super-threshold computing to NTC, alters many of the device characteristics, jeopardizing trustworthy computing. For example, a slight variation in the supply voltage impacts the circuit delay characteristics significantly, enabling copious opportunities for timing fault attacks at NTC (Section 5.2). In order to exploit this security loophole, a novel threat model is proposed, referred to as ***Ti**ming Fault **A**ttack at **N**TC (TITAN).* TITAN can inflict a range of security breaches in a low-power mobile platform.

Demonstrating the resilience to a TITAN in a real hardware setup presents a massive methodological challenge, primarily due to the unavailability of commercial NTC commodities. This challenge is tackled by architecting an ex-situ fault-attack platform, integrated with an open-source microprocessor, operating at a very low supply voltage (Section 5.4).

Using an off-the-shelf hardware environment, glitches are injected in the supply voltage of the microprocessor, running security critical applications. It is shown that two parameters of the voltage glitch, viz., *minimum glitch width* and *minimum glitch magnitude* (defined in Section 5.4.3), dictate the resilience of a system to timing fault attacks. With a fine-grain glitch control mechanism, a reduction of 1.6X (in terms of minimum glitch width), and 2.8X (in terms of minimum glitch magnitude) in timing fault resilience are demonstrated at a low-power operating condition, compared to a traditional STC operation. These results portend a tremendous susceptibility of the NTC systems to TITAN.

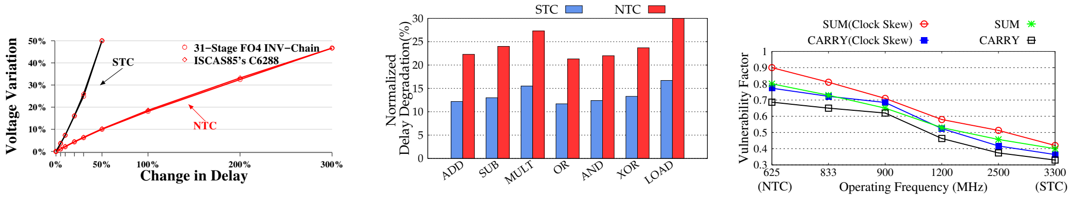
The following are the key contributions of this chapter:

5.1.1 Contributions

- The trend in the timing fault vulnerability of a given circuit is demonstrated as the circuit's operating condition shifts from STC to NTC (Section 5.2).
- The proposed threat model TITAN exploits the security vulnerabilities of NTC, to inflict potent timing fault attacks in low-power mobile platforms (Section 5.3).
- Using an off-the-shelf hardware and a voltage (VCC) glitching technique, it is demonstrated that a low-power operation exhibits an aggravated vulnerability to TITAN, compared to traditional STC (Section 5.6.2 and 5.6.3).
- By instrumenting GDB driven architected states, the user-level impacts of TITAN are analyzed in various real-life applications (Section 5.6.4).

5.2 Motivation

To establish a deeper comprehension into this research, the following two key questions need to be answered: *(a) How does NTC transform the vulnerabilities of a given circuit?* and



(a) Delay sensitivity to supply voltage variation.

(b) PV-induced delay degradation at STC and NTC, normalized w.r.t. clock period.

(c) Variation of timing fault vulnerability with clock frequency.

Fig. 5.1: Figure 5.1(a) shows a higher delay sensitivity to supply voltage variation at NTC, with respect to STC. This trend results in a higher instruction level delay degradation at NTC, compared to STC (Figure 5.1(b)). Figure 5.1(c) depicts the increasing timing fault vulnerabilities of the SUM and CARRY bits of a full-adder circuit as the operating frequency changes from STC to NTC.

(b) What are the ramifications of the circuit layer vulnerability of near-threshold computing, at the application layer?

5.2.1 Timing Fault Vulnerability: STC vs. NTC

The vulnerability of a circuit to timing faults primarily depends on the circuit's delay sensitivity to supply voltage variation. Figure 5.1(a) demonstrates a maximum of ∼6X difference in the delay variation for a couple of representative benchmarks (viz., 31-stage FO4 inverter-chain and ISCAS85's C6288) at NTC, compared to STC, under the same percentage voltage variation. The following section discusses how the process variation (PV) acts as an accomplice to aggravate this delay sensitivity, thereby exacerbating the timing fault resilience of the NTC circuits. Subsequently, the timing fault vulnerabilities of STC and NTC circuits are quantified and compared.

Process Variation

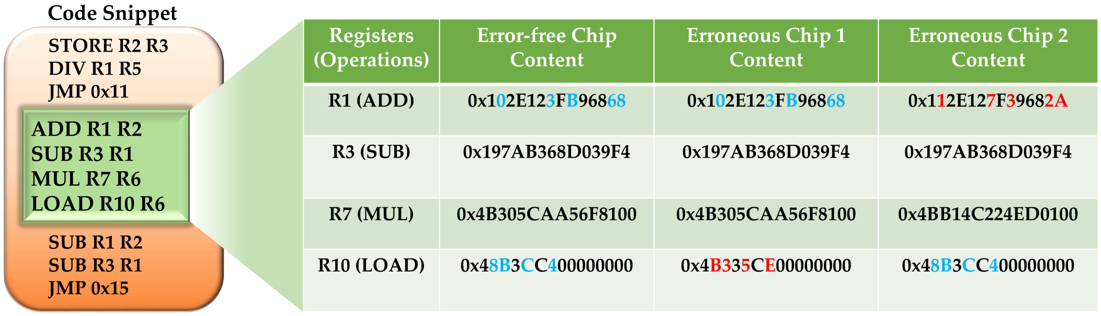
An integrated circuit operating at NTC, suffers from substantial PV-induced delay variations, up to 20X more than circuits at the nominal STC voltage [68, 80]. This delay variation can be attributed to the radically diminished voltage overdrive (i.e., the difference between the threshold and the supply voltage). In fact, a surprisingly small number of PV affected gates can drastically alter the critical paths manifested at runtime [81]. At the architecture level, a substantial variation in the instruction level delay profile is expected, seen during a program execution. Unavailability of off-the-shelf NTC commodities makes it infeasible to compare the PV-induced delay variations in fabricated STC and NTC chips. Hence, to investigate the instruction level delays at STC and NTC, a statistical timing analysis is performed, as detailed next.

Methodology

First, the execute stage of a 32-bit RISC processor with MIPS style ISA is generated, using the FabScalar setup [82]. Second, the RTL of the execute stage is synthesized with the 15-nm Nangate library [83], using Synopsys Design Compiler. The operating voltage-frequency values for STC and NTC are chosen to be (0.8V, 1GHz) and (0.45V, 250MHz), respectively. Third, using an in-house statistical timing analysis tool, the cycle accurate delay timings of all the sensitized paths of the execute stage are studied, for 7 different arithmetic and logic operations. The operands are chosen to cover a typical range seen in real applications. To model within-die process-variation at STC and NTC, the VARIUS [84] and VARIUS-NTV [68] setup are chosen, respectively.

Results

Figure 5.1(b) shows the variation in the instruction level delay—during the *execute* pipe stage—at STC and NTC conditions. The figure shows the *Normalized Delay Degradation* (Y-axis), which estimates the increase in delay, expressed as a percentage of the clock period, due to *a very small fraction of PV-affected logic gates* in the sensitized paths of an instruction. For a conservative estimate, only 0.2% gates of the circuit are assumed to be affected by PV, in all the cases. From Figure 5.1(b), two key observations can be made: (a) the delay degradation of all the instructions increase from STC to NTC, and (b) different instructions exhibit different variations in the delay profile. *Instructions with higher degraded delays are more vulnerable to PV-induced timing faults.* For example, the delay degradation of the LOAD instruction at NTC is ∼1.8X compared to that at STC. Hence, the probability of timing errors in the sensitized paths of the LOAD instruction almost doubles at NTC, with respect to STC.



Code Snippet

```
STORE R2 R3
DIV R1 R5
JMP 0x11
ADD R1 R2
SUB R3 R1
MUL R7 R6
LOAD R10 R6
SUB R1 R2
SUB R3 R1
JMP 0x15
```

Registers (Operations)	Error-free Chip Content	Erroneous Chip 1 Content	Erroneous Chip 2 Content
R1 (ADD)	0x102E123FB96868	0x102E123FB96868	0x112E127F39682A
R3 (SUB)	0x197AB368D039F4	0x197AB368D039F4	0x197AB368D039F4
R7 (MUL)	0x4B305CAA56F8100	0x4B305CAA56F8100	0x4BB14C224ED0100
R10 (LOAD)	0x48B3CC400000000	0x4B335CE00000000	0x48B3CC400000000

Fig. 5.2: Manifestation of vulnerabilities in the low-power NTC circuit at the application level. Due to the diverse impacts of process variation in three chips, clear differences in the program outcome can be observed from the highlighted code snippet. The blue and red colors signify correct and incorrect values, respectively.

An increase in instruction delay adversely impacts the vulnerability at the circuit layer. Figure 5.1(c) shows a massive increase in the *timing fault vulnerability*[1] of the primary outputs of a full-adder circuit, as its operating frequency spans from the conventional STC to the NTC regime (i.e., lower frequency). The vulnerability further aggravates in the presence of clock skew in the circuit. For this illustrative example, *Scope of Vulnerability* is used as a measure of vulnerability from timing faults [31]. *This work exploits this aggravated vulnerability of the NTC circuits to create a unique threat model based on timing fault attacks.*

5.2.2 Application Behavior under Timing Faults

Figure 5.2 demonstrates how a circuit vulnerability can have a real impact during program execution. It can be observed that an incorrect data is latched into the register file, as a snippet of code suffers from timing faults. To further complicate these issues, at NTC, the PV signature, and its corresponding delay characteristics, vary substantially among different post-silicon chips. For example, in Figure 5.2, chip 1 experiences a timing fault in the LOAD instruction, whereas, chip 2 experiences a timing fault in the ADD instruction. Such a variation can create a *moving target* for the attackers, allowing a rogue application to inflict damage, while remaining undercover due to its non-determinism.

[1] The probability of a timing violation at the output node of a circuit.

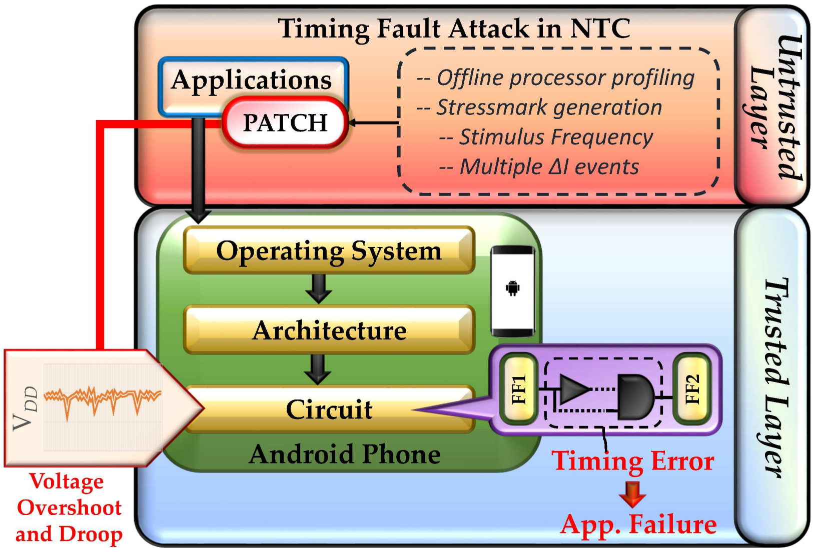


Fig. 5.3: Proposed TITAN threat model showing the trusted and untrusted components in a low-power mobile platform. A malicious patch comprising a potent stressmark induces intermittent voltage droop in the power delivery network of the processor. As a consequence, co-executing instructions of the victim application become prone to timing errors, potentially leading to discrepant behavior at the software/system level.

5.2.3 Unique Security Threat at NTC

Collectively, these trends indicate a compelling domain of research for trustworthy computing in low-power platforms. In this pursuit, *it is critical to explore a threat model that can effectively exploit the circuit-level vulnerability at the application layer in low-power systems*. Such a threat model embodies a software-hardware coalition and is effective in the modern software ecosystem.

5.3 Proposed Threat Model: TITAN

Figure 5.3 illustrates the proposed threat model—TITAN—in the realm of a low-power mobile platform, and shows the boundary of trust in the threat model. A covert insertion of a malicious patch (also referred to as a *malware*) is considered, in the application software, through a rogue application developer. The malware is essentially a potent *stressmark* that can be designed for a specific processor and is capable of sporadically generating customized voltage droop events [85]. As a consequence, the instructions belonging to the co-executing applications are susceptible to droop induced timing faults, potentially leading to erroneous and/or undesired application behaviors.

Please note that evaluating the efficacy of TITAN in real hardware is contingent on an accurate measurement of the *potency* of the malware. Given the unavailability of commercial NTC mobile processors, this challenging task is addressed by designing an ingenious hardware setup, discussed in Section 5.4. Various phases of the TITAN are described next.

5.3.1 Phases of the TITAN

To show the practical relevance of the attack model, three operational phases are formulated: *malware injection* (Section 5.3.1), *activation* (Section 5.3.1) and *operation* (Section 5.3.1).

Malware Injection

Due to a rigorous process of software testing, it is assumed that the initial release of the application software is free from a malware. Subsequently, the software can be tampered through a malicious patch [86]. The malware, having an extremely low activity footprint, can be easily camouflaged in the statistics collection module of the software, which aims to improve the user experience [87]. The modified application can then initiate its operation (detailed in Section 5.3.1), disrupting trustworthy computing in an NTC processor. The malpractice of downloading updates from untrusted repositories, magnifies the scope of a malware injection using this route.

Activation

The malware can be launched using an automatic trigger-based activation technique, which is very challenging to detect [88]. An existing bug in the legacy code can trigger the malware by creating a rare event [89]. To ensure a stealthy operation, a TITAN creates a time-based trigger, similar to the interrupt handler of a watchdog timer. The trigger can be implemented as a conditional jump, evaluating on the rare event. The attack vector is distributed to prevent an early detection.

Operation

The malware uses software-induced fault generation technique to launch potent timing fault attacks. In this pursuit, the malware is instrumented with *fault inducing instructions*, to impact the execution of the *vulnerable instructions* of the victim application. Each of these distinct categories of instructions is described next.

Fault Inducing Instructions: Researchers have discovered that when a sequence of high power consuming instructions is executed immediately after a sequence of low power consuming instructions, a local voltage droop (i.e., an inductive or a ΔI droop event) is induced in the power delivery network (PDN) of the core [90]. Collectively, these two sequences of instructions are called a *stressmark*. Zhang et al. have recently shown that the NTC systems are highly vulnerable to inductive droop, despite being dominated by a large leakage current [91]. This drawback is exploited to create a stressmark in the threat model which is capable of overwhelming the design margin of the NTC processors by inducing a large voltage droop in the PDN of the core.

This induced voltage droop can increase the instruction delays, potentially causing timing faults in the datapaths between a pair of pipeline register banks [92]. Since, the delay sensitivity to supply voltage variation is much greater at NTC, compared to STC (due to an aggravated impact of PV), the execution of a stressmark makes some of the co-executing *benign* instructions in the pipeline vulnerable to timing errors.

Vulnerable Instructions: Section 5.2 demonstrates how each fabricated NTC chip of the same design can have a different set of long delay instructions, post fabrication, due to PV. As these long delay instructions are highly susceptible to timing fault attacks, they are termed as the vulnerable instructions. *The malware aims to induce timing faults in the majority of these vulnerable instructions, belonging to a co-executing benign application.* Due to a substantially varied PV signatures of the fabricated NTC chips, the same malware in different chips can potentially cause timing faults in different vulnerable instructions of the same application. Such non-determinism allows TITANs to inflict a range of end-user breaches in a stealth mode.

5.3.2 Scope of a TITAN

The scope of a TITAN depends on the specific vulnerabilities of a chip. A timing fault in an issue stage, may cause a premature dispatch of an instruction, leading to a *data hazard* in the pipeline. On the other hand, repeated fault attacks in a branch instruction, may always result in a *taken branch*. Such architectural events can have different ramifications at the application layer (Section 5.6.4).

5.3.3 Limitations of a TITAN

- It is assumed that the processor cores of an NTC system support the Hyper-Threading technology, so that multiple applications can simultaneously execute as different independent threads, belonging to the same core. Consequently, the malware, executing on one thread of a core, can launch timing fault attacks on a benign application, executing on a different thread of the same core.

- The malware is agnostic of the timing characteristics of the victim applications. During the intermittent active phases (Section 5.3.1), the malware aims to induce timing faults in the vulnerable instructions of all the co-executing benign applications. As a result, an end-user is likely to witness different and unpredictable faulty application behaviors at different times, due to the same TITAN malware.

- In a multi-core system, if the malware and the victim application execute on different cores, the former is unlikely to launch a successful timing fault attack on the latter. This is because, the effect of the malware-generated voltage drop is inversely proportional to the distance between the cores [93]. Moreover, modern multi-core devices dedicate a separate power delivery network and power management policy for each core, thereby further reducing the impact of voltage drop from one core to the other [94]. To efficiently tackle these challenges, an adversary can set the *CPU affinity mask* of the malware to multiple core IDs of the device [95]. Therefore, the malware can sporadically run on different cores, increasing the efficacy of TITAN.

5.3.4 Can Rebooting the Device Tackle TITAN?

Manually rebooting a device affected by TITAN can temporarily avoid the imposed security threat. However, the device will be exposed to the same threat upon restart; or it might have already been compromised even before rebooting. Moreover, rebooting may not be a plausible option for many emerging low-power IoT devices. For example, health monitoring devices can risk the loss of a patient's life if it undergoes an untimely reboot [96]. Also, a smartphone acting as a data aggregator in a remote patient monitoring system based on an IoT cloud architecture [79], may not have the opportunity for rebooting, lest affected by a TITAN.

5.3.5 Classic Malware Vs. Malware in TITAN

The extent of the damage inflicted by most existing malware is limited to the software level. Moreover, most of such malware can be efficiently detected and terminated by available anti-virus software. On the other hand, the malware in TITAN, exploits a circuit-level vulnerability (i.e., a high delay sensitivity to supply voltage variation, resulting in a poor tolerance to timing fault attacks), to tamper application/system level integrity. Moreover, the vulnerabilities of the hardware being widely varying across various chips of a given design (due to PV), the same malware in TITAN can potentially cause different and unpredictable application malfunctions in different chips of the same design (Section 5.2.2 and 5.3.2). Consequently, designing a generic detection and prevention scheme for TITAN is extremely challenging.

5.4 Hardware Setup to Gauge Resilience Against a TITAN

In this section, the proposed hardware setup is discussed to evaluate the resilience of the low-power systems against a TITAN.

5.4.1 Unavailability of NTC Hardware

Designing an actual TITAN system is practically very challenging, primarily due to the unavailability of off-the-shelf NTC mobile commodities. Intel demonstrated a proof-of-

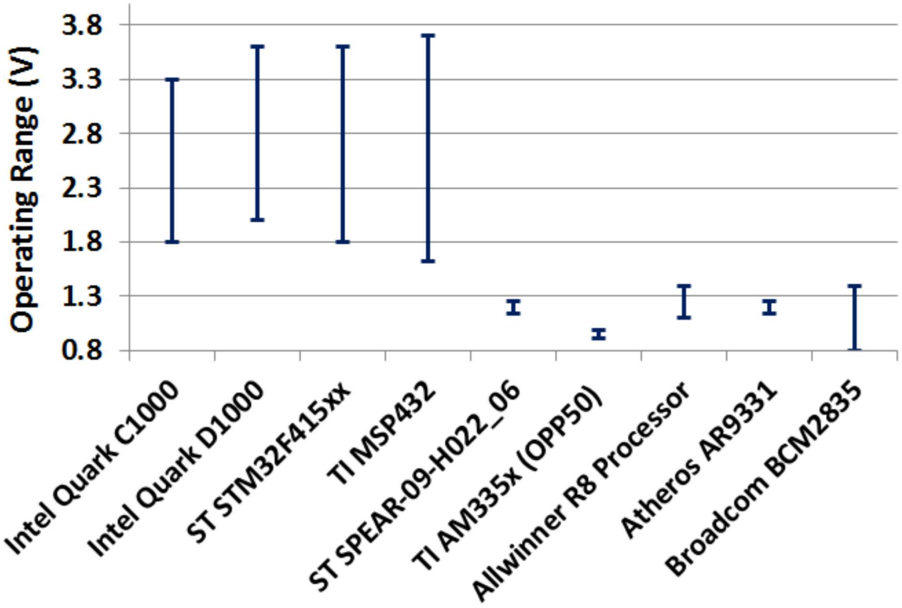


Fig. 5.4: Operating voltage ranges of state-of-the-art commercial low-power processors.

concept Near-Threshold Voltage x86 microprocessor, called Claremont, at ISSCC 2012 [97]. However, Claremont is not available in the market for purchase. On the other hand, the state-of-the-art low-power processors operate in the STC range. Figure 5.4 shows the operating voltage ranges of some commercially available low-power processors. The lower limit of the operating voltages among these chips is the lowest (0.8V) for *Broadcom BCM2835*, based on the Raspberry-Pi official documentation [98]. *Hence, Raspberry-Pi is chosen as the platform for realizing TITAN.* Please note that existing works on low-voltage fault attacks also could not operate a general purpose processor below 0.9V [99].

Minimum Operating Voltage: Although the Raspberry-Pi official documentation states that the core can be undervolted up to 0.8V by changing the voltage-frequency (VF) settings in the OS configuration files, the actual voltage measured at the V_{dd}_core pin [100], does not fall below 1.05V. It is also observed that despite switching the V_{dd}_core supply to an off-board regulated power supply after successfully booting the core with the on-board supply, the core stops working below 1.05V. Hence, 1.05V is the minimum operating voltage in the experiments. However, 1.05V does not truly represent an NTC operating voltage. Considering this practical limitation, the fault resilience of the Raspberry-Pi is evaluated as the operating voltage sweeps from 1.4V to 1.05V. Based on the resilience trend in that voltage interval, the fault resilience at NTC operating conditions is predicted (Section 5.6).

5.4.2 Malware Emulation

The malware in TITAN is a potent stressmark that induces timing faults by occasionally introducing voltage droop in the power supply of the underlying processor (Section 5.3). To gauge the resilience of a processor to fault attacks at various VF levels, one needs to generate stressmarks of precise magnitude and duration. Automating custom stressmark generation, although explored by many researchers in the past [101], requires a rigorous cross-layer methodology which is orthogonal to this work. Moreover, during a low-voltage operation, a large voltage droop (beyond the guardband of the processor) can potentially disrupt the OS kernel, resulting in a failure to record the precise stressmark parameters, if the malware is running on the victim processor itself.

Hence, to mimic the behavior of the malware in TITAN, a fine-grain voltage (VCC) glitch injection technique is used that launches timing faults (via a separate hardware setup) in the victim processor. This is a reasonable approach, as both voltage droop and VCC glitch can manifest as timing faults in an integrated circuit, despite having different temporal characteristics. As the primary goal of the hardware experiments is to demonstrate the resilience of a system against a TITAN, the source of timing faults (be it stressmark induced voltage droop or VCC glitch) does not have a paramount impact on the final conclusion (Section 5.6).

Next, two important glitch parameters are discussed to gauge the resilience of a system to TITAN (Section 5.4.3). Subsequently, the roles of each hardware component are described in relation to a TITAN (Section 5.4.4).

5.4.3 Important Glitch Parameters

Figure 5.5 depicts the two crucial glitch parameters, *glitch width* and *glitch magnitude*, that are studied in this work.

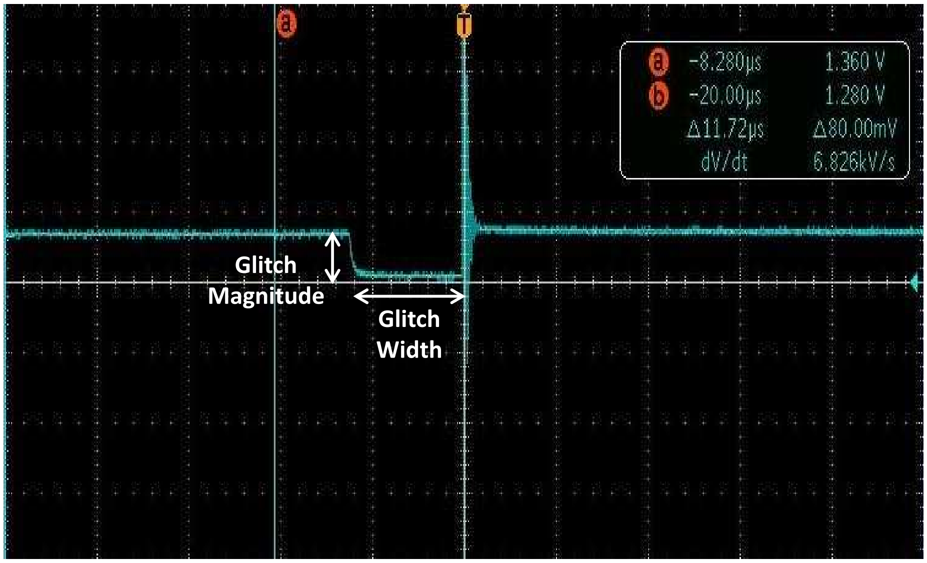


Fig. 5.5: Important VCC glitch parameters: glitch width and glitch magnitude.

Glitch Width

Glitch width is the time duration for which a voltage glitch lasts. A relatively larger glitch width is equivalent to a relatively large number of consecutive ΔI droop events, created by a potent stressmark. For a given system, running at a specific operating voltage-frequency, *minimum glitch width* is defined as the shortest time duration of a voltage glitch, sufficient to cause the Point of First Failure (PoFF) [102] in a running application. Hence, *greater the minimum glitch width, more resilient is the system to timing fault attacks.*

Glitch Magnitude

Glitch magnitude is the absolute amount of drop in the supply voltage from its nominal value. A larger glitch magnitude is analogous to a larger ΔI voltage drop. *Minimum glitch magnitude* is defined for a system operating at a given voltage-frequency, as the smallest absolute drop from the nominal supply voltage, sufficient to cause the PoFF in a running application. Hence, like minimum glitch width, *greater the minimum glitch magnitude, more resilient is the system to timing fault attacks.*

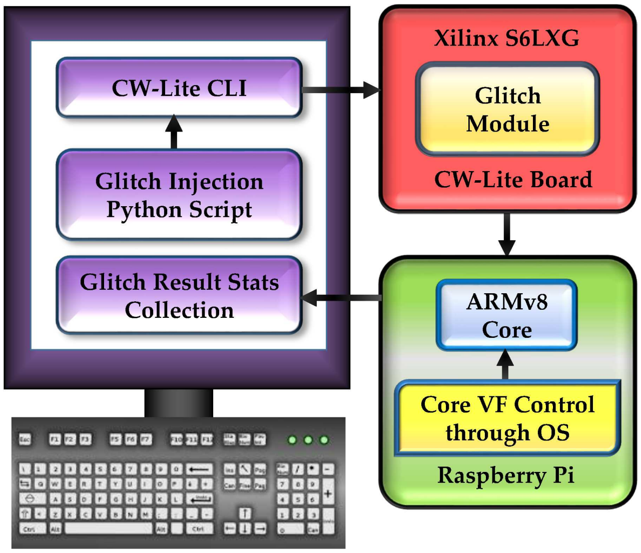

CW-Lite CLI
Glitch Injection Python Script
Glitch Result Stats Collection
Xilinx S6LXG
Glitch Module
CW-Lite Board
ARMv8 Core
Core VF Control through OS
Raspberry Pi

Fig. 5.6: Conceptual block diagram of the TITAN hardware environment. The ChipWhisperer-Lite (CW-Lite) board acts as an attacker that launches fault attacks in an application running on the victim Raspberry-Pi module.

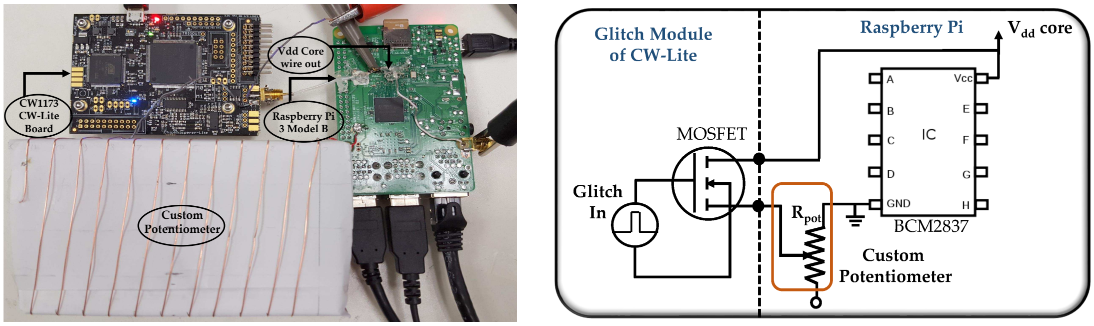
5.4.4 Hardware Components

Figure 5.6 shows the conceptual overview of the experimental setup. To launch a timing fault attack, the VCC glitch injection module, embedded in the *ChipWhisperer-Lite board* is used. The glitch injection module injects glitches in the supply voltage of the victim *Raspberry-Pi* that runs an application software. The ChipWhisperer-Lite board and the Raspberry-Pi are interfaced with a desktop computer, used for triggering glitches, controlling various glitch parameters and collecting glitch results. The component details of the attacker and the victim modules are discussed next.

ChipWhisperer-Lite: The Attacker

Figure 5.7(a) shows the CW1173 ChipWhisperer-Lite (CW-Lite) board.

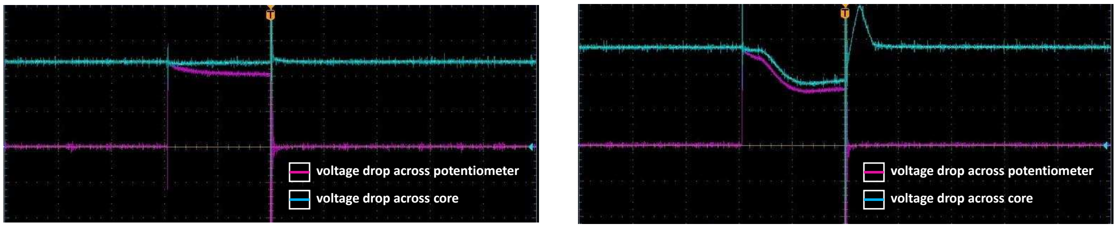
Glitch Injection Module: The primary component of the module is a Xilinx S6LX9 FPGA. The Digital Clock Manager (DCM) block inside the FPGA is used to configure various glitch parameters at the design time. However, using Xilinx's Partial Reconfiguration (PR) technique, multiple adjustments can be performed at runtime by instrumenting the partial bitstreams. This PR technique is exploited to control the glitch parameters.



(a) A ChipWhisperer-Lite board connected to a Raspberry-Pi. The magnitude of the injected glitch is controlled using a custom potentiometer.

(b) VCC glitch module of the CW1173 ChipWhisperer-Lite board, connected to the BCM2837 chip of the Raspberry-Pi.

Fig. 5.7: TITAN hardware environment and the circuit diagram for VCC-glitch injection.



(a) Voltage drops across the core and the potentiometer during a failed fault attack at (1.2V, 700MHz).

(b) Voltage drops across the core and the potentiometer during a successful fault attack at (1.4V, 800MHz).

Fig. 5.8: While triggering a glitch, a larger voltage drop across the potentiometer, due to including all the turns of wire, results in a very small voltage drop across the core. So, the fault attack is not successful. On the other hand, if a fewer turns of wire is included in the potentiometer, the core input voltage drops appreciably, resulting in a successful fault attack.

Controlling Glitches: At each VF level of the victim hardware, running an application, the *minimum glitch width* and the *minimum glitch magnitude* are ascertained (Section 5.4.3). A constant glitch magnitude (equals to the victim supply voltage: *vdd core*) is maintained while ascertaining the *minimum glitch width.* On the other hand, the largest possible glitch width from the module is maintained while determining the *minimum glitch magnitude.*

- For each round of experiment, a custom automation script is used to trigger a glitch 100 times, with an interval of 500 milliseconds between two consecutive glitches. A faster triggering rate might damage the glitching MOSFET of the DCM (used to short the victim core's power line to the ground voltage) due to a high frequency short circuit current through it. On the other hand, a slower triggering rate will prolong the turnaround time of the experiment.

- The glitch injection module allows customization of the glitch width. However, the module only glitches the victim's supply voltage to the ground (∼0V), and does not allow controlling the glitch magnitude. The voltage at the victim core during the glitch, depends on the length of the wire connecting the CW-Lite board and the victim (Figure 5.7(b)). Hence, the glitch magnitude is varied by changing the length of that wire using a custom potentiometer, connected between the grounds of the glitch module and the victim core ($R_{pot.}$ in Figure 5.7(b)).

- Figure 5.8 depicts two screenshots of the oscilloscope signals, during a failed and a successful fault attack, respectively. In the first case, most of the voltage is dropped across the potentiometer due to including all the turns of wire, and hence the core supply voltage does not drop enough to cause a successful fault attack. However, in the second case, the core supply voltage drops sufficiently due to a reduced wire-length of the potentiometer, resulting in a successful fault attack.

- It is confirmed that the *minimum glitch width* and *minimum glitch magnitude*, measured at each voltage-frequency, reflect the resilience of only the processor core of the Raspberry-Pi, and not of any other parts of the system. In this pursuit, only the core VF-level is reduced, and the memory VF-level is kept constant throughout the experiments, to realize a realistic NTC system [81]. Also, the activity of the I/O is kept as minimum as practically possible, for example, by doing SSH through Ethernet over HDMI, to connect to the Raspberry-Pi, as the former is more resistant to transient voltage fluctuations during glitch injection, and has a negligible activity footprint.

Raspberry-Pi: The Victim

Raspberry-Pi 3 Model B is used as the target victim platform on which a VCC glitch attack can be launched during an application execution. This model has a 1.2 GHz 64-bit quad-core ARMv8 CPU with 1 GB RAM and VideoCore IV 3D graphics core, and runs on

Raspbian OS. The input voltage port of the ARM core is connected to the glitch port of the CW-Lite board (Figure 5.7(a)). In the following, the on-board voltage-frequency control mechanism is discussed to create various operating points.

On-board Voltage-Frequency Control: The Raspbian OS allows sweeping the operating voltage of the BCM2837 chip (ARMv8 core) with a granularity of 25 millivolt. The operating frequency can be controlled through a system file.

- The operating conditions are varied until the lowest allowable VF-level of the Raspberry-Pi. For the experiments, the range of operating conditions is: (1.4V, 800MHz) to (1.05V, 600MHz). The experimental results, obtained from this operating range, portends an extremely aggravated vulnerability of the NTC systems to TITAN, compared to traditional STC systems (Section 5.6).

- To ensure sufficient available slack in the system, the operating frequency is reduced by 50MHz, for a reduction of ∼0.1V in the operating voltage. It is also ensured that each application runs without any error when no glitch is applied.

5.5 Applications

In this section, a brief description of the applications is given that are used to evaluate the resilience against a TITAN at various operating conditions.

- **Edge Detection App:** This application is based on a water flow-like metaheuristic algorithm in Python. The algorithm emulates the behavior of water that always flows down to lower altitude regions and divides regions based on the minima that water approaches.

- **Matrix Multiplication:** This application performs 8-bit matrix multiplication in C++ using Eigen APIs.

- **K-means Clustering:** The application uses an algorithm, implemented in C, that is based on a single-pass implementation with an iterative batch update process. The error in application execution is measured in terms of erroneous cluster membership.

- **Decision Tree Classifier:** The application implements an entropy-based decision tree binary classifier in C++. The accuracy of the algorithm is determined based on 10-fold cross validation on a multivariate dataset from the UCI Machine Learning Repository.

5.6 Experimental Results

In this section, the results of the TITAN hardware experiments are analyzed. Section 5.6.1 describes the system level impacts of a successful timing fault attack in the experiments. Section 5.6.2 and 5.6.3 present the variation of *minimum glitch width* and *minimum glitch magnitude* at different operating conditions, revealing the trend of timing fault resilience. Section 5.6.4 presents the tangible impacts of TITAN in a few real applications.

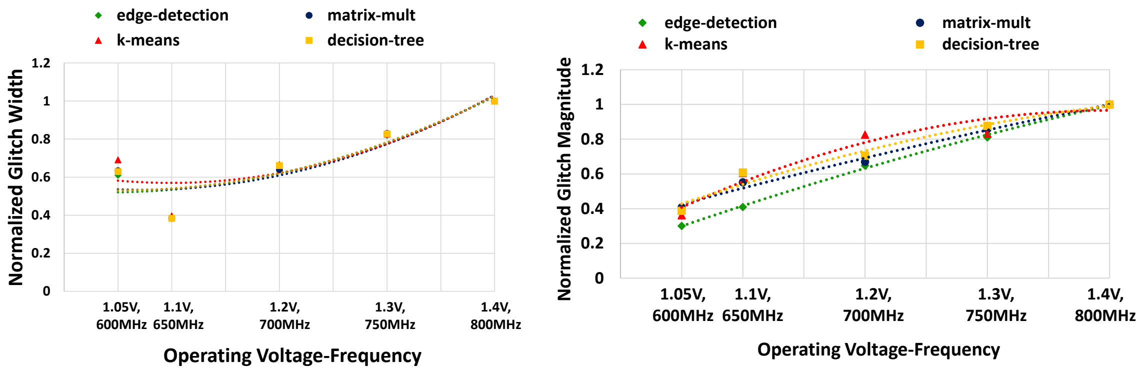
5.6.1 System Level Impacts of Fault-Attacks

In the experiments, a successful timing fault attack results in one of the following occurrences: segmentation fault in the application, crash in the operating system, kernel error or system failure. Such a deviation from the expected operation of an application can have serious consequences. For example, a successful fault attack in an edge detection application used in a pattern based password recognition system, can deny a sign-in process despite drawing the correct pattern.

5.6.2 Minimum Glitch Width Variation

Figure 5.9(a) shows the variation of *minimum glitch width*, with operating voltage-frequency for four applications. The Y-axis values are normalized to the *minimum glitch width* at the highest operating point (HOP): (1.4V, 800 MHz).

The trend-lines for all the applications exhibit a monotonic decrease in the *minimum glitch width* as the operating condition shifts towards the NTC values. Two crucial conclusions are drawn from Figure 5.9(a). First, the lowest operating point is significantly more susceptible to timing faults (1.6X on an average) with respect to the HOP, for all the applications. Second, a rise in the *minimum glitch width* is observed from (1.1V, 650 MHz) to



(a) Variation in minimum glitch width.

(b) Variation in minimum glitch magnitude.

Fig. 5.9: Variation in minimum glitch width and minimum glitch magnitude. A lower value in the Y-axis indicates a poorer fault resilience.

(1.05V, 600 MHz). This apparent anomalous behavior is due to a higher available slack in the system, caused by an incommensurate reduction of the supply voltage, with respect to the reduction in the operating frequency. However, despite the availability of more timing slack, a lower operating point exhibits a poor fault tolerance in general, primarily due to an increased instruction level delay sensitivity to supply voltage variation. As an NTC system operates at a much lower VF-level than the operating points in the experiments, it is conjectured that the timing fault resilience at NTC is much worse, leading to a significantly increased vulnerability to TITAN.

5.6.3 Minimum Glitch Magnitude Variation

Figure 5.9(b) shows the variation in *minimum glitch magnitude* with operating voltage-frequency for four applications. The Y-axis values are normalized to the *minimum glitch magnitude* at the HOP (Section 5.6.2).

It can be noticed that the trend-lines for all the applications monotonically decrease from HOP towards the lower operating points, reflecting a diminished fault tolerance with decreasing voltage-frequency. On an average, the lowest operating point in the experiment is 2.8X more vulnerable to timing fault attacks, compared to the HOP, considering all the applications. In the experiments, it is observed that a glitch magnitude, as low as 10% of the supply voltage, can cause a successful fault at the lowest operating point. These results lead to a conclusion that a relatively weak stressmark is potent enough to inflict visible faulty behaviors during an application execution at the NTC operating conditions. Employing a weak stressmark makes TITAN more stealthy due to an insignificant side-channel. Also, the drastic impact of PV would likely further degrade the NTC systems' fault resilience, thwarting secure operations.

5.6.4 Practical Examples of TITAN

The potential impacts of TITAN are presented in a few real applications, *assuming a majority of the vulnerable instructions in an application encounter timing faults due to a TITAN*. While such timing fault attacks can be performed at STC operating conditions too, the high impact of PV makes NTC systems particularly susceptible to TITAN (Section 5.2.1). In the following discussion, the first four applications are part of the SPEC CPU 2006 benchmark suite [103]. The results are obtained by instrumenting the GDB driven architected states during the application execution.

- ***Corrupted Uncompressed Data in Bzip2:*** Assuming $BRANCH$ being the vulnerable instruction, faults are injected in the branches of the uncompress-stream module of *Bzip2*, which can corrupt the data that is required to get the logical end of stream. As a result, the uncompressed data fails to validate data integrity, as indicated by the CRC flag of the program.

- ***Failed Factorization in Libquantum:*** The Greatest Common Divisor (GCD) unit in the Shor's factorization module of *Libquantum*, has multiple calls to $BRANCH$ instructions. Considering $BRANCH$ to be the vulnerable instruction, faults are introduced in such branches which can lead to an incorrect inference (e.g., branch *not taken*, as opposed to *taken*), potentially causing an error in the GCD calculation. The incorrect GCD computation results in a failed factorization for a composite number.

- ***Sub-optimal Vehicle Schedule in Mcf:*** The main *Mcf* engine uses a primal simplex method to obtain the optimal vehicle schedule. It creates artificial arcs for each node. Each arc has a positional attribute with some values. Based on certain conditions, the position of each arc is updated. Once again, considering $BRANCH$ as the vulnerable instruction, faults are injected in them, so that the program can erroneously bypass the check for the aforementioned conditions, resulting in an incorrect arc position. Such a fault leads to a sub-optimal vehicle schedule, as indicated by the checksum difference between the fault-free and the faulty execution of the program.

- ***Erroneous Force Calculation in Namd:*** The computational core in *Namd* calculates the movement of the atoms for the patches in the input patch list. Assuming $LOAD$ to be the vulnerable instruction, faults are introduced in them, and as a result, the position of a given atom in the patchlist may not be updated. Failing to update the position of an atom results in a miscalculation of the force, which is reflected by an incorrect checksum with respect to a baseline execution of the application with no timing fault.

- ***Illegal Queen Configuration in N-Queens:*** This benchmark implements a propagation and backtracking based approach to solve the popular N-queens problem. The implemented algorithm finds a single legal queen configuration with polynomial time complexity [104]. The safe check module of the benchmark checks if a queen can be placed at given row-column pair of the board. If $LOAD$ is assumed to be the vulnerable instruction, injecting faults in them can cause an incorrect inference by the safe check module. As a result, it returns an illegal board position as a valid configuration for the game.

CHAPTER 6

EXPLORING A FOCALLY INDUCED FAULT ATTACK STRATEGY IN NEAR-THRESHOLD COMPUTING

6.1 Background and Contributions of This Work

Computing is on an upward trend of integrating with humans and our environment through an interconnected network of wearables, digital dust, smartphones, and desktops. With this upsurge, two fundamental trends in low power computing are observed that are poised to reshape the trustworthy properties of future hardware platforms. First, there is a dramatic rise of low power systems and applications (e.g., Internet of Things (IoT)) that are systematically intertwined into our mere existence. Many IoT systems, implantable medical sensors, for example, have stringent constraints on power consumption, while demanding ever growing computing power and functional diversity. Second, hardware researchers are actively engaged in exploring a new generation of device and circuit technologies that can offer massive improvements in the energy efficiency of computation. In this ever-growing demand for energy efficiency, *Near-Threshold Computing (NTC)* has emerged as one of the promising directions [3]. While many recent works have explored the challenges and opportunities of reliable computing at near-threshold [91], the *security vulnerabilities* spawning from such low-power computing frameworks have received only a marginal attention.

NTC circuits feature a tremendous delay sensitivity to supply voltage variation [68], enabling ample opportunities for timing fault attacks [31]. Recently, Bal et al. have shown that a small number of process variation (PV) affected logic-gates at NTC, can alter the critical paths, sensitized during a program execution, which was unprecedented in the realm of traditional super-threshold computing [105]. Moreover, the PV-induced delay variability is substantially different in various NTC chips of the same design [68]. Considering these trends, a threat model is proposed, referred to as a *Focally Induced Fault Attack (FIFA)*, that exploits the circuit-level variability of the NTC systems, to inflict security threats at the application-level. FIFA employs a machine learning framework to design a malicious software module. The malicious module, analogous to a polymorphic virus, learns about the circuit-level vulnerabilities of individual victims—NTC processors—and subsequently uses the knowledge to launch targeted fault attacks. Specifically, the following are the key contributions of this chapter.

- A novel threat model, called FIFA, is proposed that exploits the emerging security vulnerabilities of the NTC circuits to inflict potent timing fault attacks in low-power mobile platforms (Section 6.2).
- A machine learning framework, called *SmartLearn*, is developed to identify the circuit-layer vulnerabilities and generate software modules to cause a breach in security (Section 6.3).
- Using the *SmartLearn* framework, the vulnerable instructions are ascertained for various NTC chips of the same design (Section 6.5).

6.2 Proposed Threat Model: FIFA

Figure 6.1 illustrates the proposed threat model, FIFA, in the realm of a low-power mobile platform. FIFA is inspired by a fundamental principle of a viral attack in a biological system, where the virus is able to mutate itself during its transmission from one human host to another, thereby vastly improving its efficacy in different hosts with diverse immunities. Figure 6.1 shows the boundary of trust in FIFA. It is considered that a rogue software developer will stealthily insert a malware in an application software. This malware learns the specific vulnerabilities of its victim processor, and subsequently morphs itself to launch potent fault attacks.

6.2.1 Phases of the FIFA

Three operational phases of FIFA are discussed in order to establish its practicality.

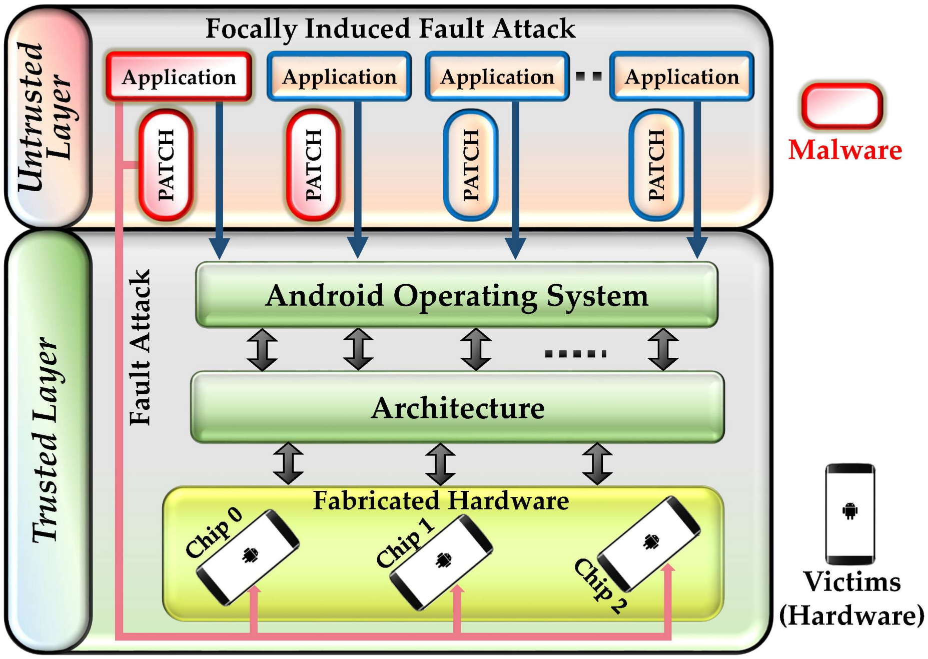


Fig. 6.1: The Proposed FIFA threat model showing the trusted and untrusted components in a low power mobile platform.

Malware Insertion

The malware can be inserted in the application software in two different ways. First, a few rogue application developers can compromise the rigor of the software testing process, and introduce a malware in the initial release of the software. The malware, with a low footprint, can efficiently disguise in the statistics collection module of the software that collects user-approved logs for offline analysis [87]. Second, the initial software release can be malware-free, but subsequently, it may be tampered through a malicious patch [86]. The modified application can then launch a FIFA, compromising the integrity of the mobile platform. Many users download and install updates from untrusted repositories, which intensifies the scope of a malware insertion through unreliable, malicious patches [106].

Activation

An existing bug in the legacy code of the application software can trigger the malware by creating a rare event [89]. A covert FIFA can be ensured by using a time-based trigger, similar to the cron job in a Unix based operating system. Moreover, the diverse PV signatures of the NTC chips and the corresponding delay characteristics prevent an early detection of the malware.

Operation

The malware adopts a two-pronged strategy: diagnose and learn the vulnerabilities of the victim processors, and launch targeted FIFAs exploiting the vulnerabilities.

- **Learning Vulnerabilities:** Different chips of the same design can have different delay profiles due to PV, signifying diverse vulnerabilities. The malware comprises a low-overhead diagnostic module that learns the vulnerable instructions of various victim processors by inducing timing faults in the victims from the software layer. The malware uses the popular software-implemented fault injection technique to induce timing faults (details in Section 6.3.3). The malware then offloads the vulnerable instruction data, consolidated from many victim processors, to a remote server. Considering multiple remote servers prolongs the eradication of the attack, thereby fortifying the threat model.
- **Launching FIFAs:** At the remote server, intelligent machine learning algorithms are used to identify a small, yet potent, set of vulnerable instructions, that can harm a large majority of the victim processors. The malware then creates a malicious patch that launches targeted FIFAs, when installed during a software update process in the victim mobile phone processors.

6.3 SmartLearn: A Cross-Site Learning Framework

Figure 6.2 illustrates different components of the proposed learning framework—*SmartLearn*. *SmartLearn* aims to automatically learn the vulnerabilities of various chips of a design, to facilitate a FIFA. Using a cross-site collaborative framework, *SmartLearn* exchanges information between the malicious application server, and the victimized client hardware. The design challenges of *SmartLearn* are outlined in Section 6.3.1, and the learning algorithm is discussed in Section 6.3.2.

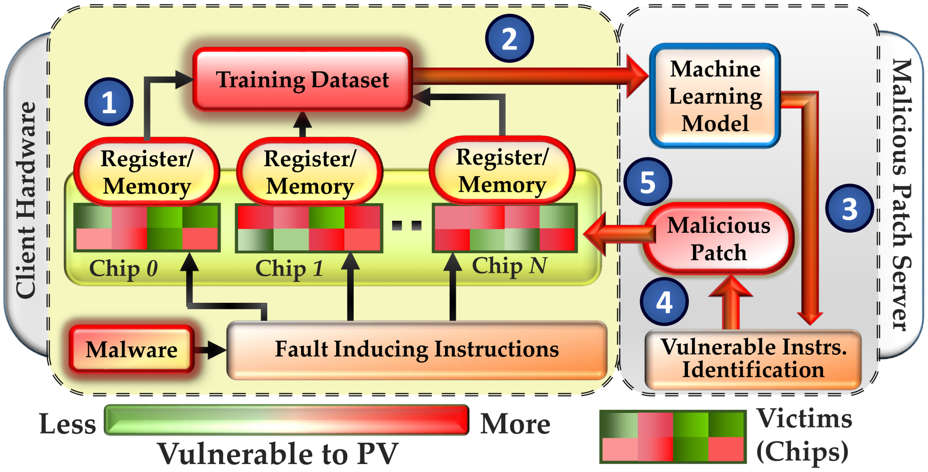


Fig. 6.2: Various operational stages of a FIFA using SmartLearn. Five distinct phases of FIFA are marked as: (1) vulnerability diagnosis at the chip level; (2) vulnerability data transfer to remote server; (3) Processing diagnostic data using SmartLearn; (4) Customized malicious patch creation; and (5) patch delivery to its target chip.

6.3.1 Challenges of SmartLearn

Creating a rogue learning infrastructure faces a couple of unique design challenges, outlined next.

Colossal Search Space

An attacker needs to retrieve the vulnerabilities of millions of chips of a design, in order to create a malicious patch, equally detrimental for a majority of the chips. As the vulnerabilities vary across the chips, designing a potent customized attack for each chip would extend the search space to mammoth proportions. Hence, it is imperative to optimize the search space for a few consolidated patches, that would be maximally damaging for the largest possible share of the chips.

Vulnerability Diagnosis

An effective FIFA design involves an accurate assessment of the hardware-layer vulnerability through the software-layer. It is extremely difficult for an application software, to precisely inject a timing fault in the underlying circuit, crucial for the vulnerability detection. Moreover, ensuring a stealthy operation requires the malicious software to maintain a low activity footprint.

6.3.2 Design of SmartLearn

To overcome the design challenges, *SmartLearn* employs a holistic *Machine Learning* approach, comprising four stages: (a) Software-Induced Fault Generation, (b) Training Data Accumulation, (c) Machine Learning Model Development, and (d) Patch Formation. Stage (a), (b) and (c) correspond to step 1, 2 and 3 in Figure 6.2, respectively. Stage (d) corresponds to step 4 and 5, in Figure 6.2. Each of these stages is described next.

6.3.3 Software-Induced Fault Generation

It is assumed that the vulnerable instructions of a (post-silicon) validated chip cannot deterministically cause timing violations. This is a reasonable assumption as timing violations are highly contingent on complex sequencing of opcodes and operands, and determining such sequences is non-trivial [107]. Hence, the malware needs to create an environment, that induces timing violations during an instruction execution. In this pursuit, the malware is instrumented with two distinct categories of instructions, described next.

Fault Inducing Instructions

Bertran et al. have shown that consecutive executions of a sequence of low power consuming instructions and a sequence of high power consuming instructions induce a voltage droop in the power delivery network of the core [90]. Collectively, these two sequences of instructions are called a *stressmark*. The induced voltage droop can increase the instruction delays, potentially causing a timing fault within a pipeline stage [92]. Due to an aggravated delay sensitivity to supply voltage variation at NTC, the execution of these fault inducing instruction sequences makes the next few *benign* instructions in the pipeline vulnerable to timing errors. Multiple distinct droop levels are considered to enforce a timing error for a given instruction (Section 6.3.4). Different levels of voltage droop can be precisely generated at the software level using well-known stressmark generation methodologies [101, 108]. As the stressmark parameters are ascertained at the remote server, the associated overheads do not impact the stealth of the malware.

Vulnerable Instructions

These instructions contain a subset of all the opcodes supported by the ISA of the victim processor. For a given instruction, the impact of the operand length on the occurrence of timing errors, is examined. To estimate the vulnerability, each opcode is paired with operands of varied lengths, and the extent of violations for such pairs are observed. *An instruction is assumed to be vulnerable, only if it latches an incorrect data in the register file, after the execution of the fault inducing instruction sequence.* Identifying the vulnerable instructions is done in multiple short intervals to maintain the stealth of the malware.

6.3.4 Training Data Accumulation

In the *SmartLearn* framework, the vulnerable instructions, along with a few metadata, comprise the features of the training data set. A few features that are studied, are: *opcodes, number of operands, input operand widths, multiple droop levels for fault injection, and the number of bit flips in the pertinent register file entry.* The malware reads the register values after the execution of the intended instruction, and compares it against the correct stored values, to infer a timing error. The violated instructions and the aforementioned metadata are stored in the application's memory space, in the form of a hash-table. Considering 34 opcodes of the ARM ISA, with each instruction being 32 bits long, and 8-bit entries to hold the number of bit-flips per instruction for 3 droop levels, the size of the hash-table is approximately 250 bytes.

6.3.5 Machine Learning Model Development

A machine learning model is developed at the server, to analyze the diagnostic information on vulnerable instructions, collected from millions of chips of a given model, and to create a few code patches that offer potent attacks to a majority of these chips. This work compares the performances of multiple machine learning algorithms viz., Support Vector Machine, XGBoost, Naive Bayes classifier and Random Forest, on the training data set.

The details on features selection, model accuracies and determination of the vulnerable instructions are discussed in Section 6.5.

6.3.6 Patch Formation

Upon building the model, its accuracy is evaluated using 10-fold cross validation. Subsequently, the opcodes are ranked based on their inferred vulnerabilities and suitable operands are determined to create the most vulnerable instructions (Section 6.5.2). These vulnerable instructions are then included in the malicious patch. The patch, when installed during a software update process, launches FIFAs in the client hardware.

Note that generating a distinct patch for every chip is not realistic in the proposed threat model. To ensure a covert operation, the malware collects and offloads only a minimum diagnostic information for each victim, to the remote server (Section 6.3.4). Deploying customized patches to the appropriate clients requires the signatures of the clients. Transmitting such signatures would incur additional overhead, thwarting the stealth of the FIFA. Hence, a single consolidated patch is generated, detrimental for a large share of the victim NTC chips.

6.4 Methodology

The PTM 14-nm technology model cards are customized for HSPICE, to study the comparative delay distributions for the basic gates at the NTC region [73]. To model within-die process-variation, the VARIUS-NTV setup is used [68]. The RTL of the execute pipe-stage of a 32-bit FabScalar generated core is synthesized [82], with the 15-nm Nangate library [83], using the Synopsys Design Compiler. The operating voltage and frequency at NTC are assumed to be 0.45V and 250MHz, respectively.

An in-house statistical timing analysis (STA) tool is used to study the timing errors of all the sensitized paths of the execute stage, for 7 different arithmetic and logic operations [105, 109]. The operands are chosen to cover a typical range seen during the executions of real applications. The STA tool uses a library of delay files for the basic gates, at different operating conditions. The library comprises both PV-affected and PV-free delay information for all the gates. The inputs to the STA tool are the synthesized netlist of a circuit and the input vectors corresponding to the netlist. The input vectors can be generated from an architectural simulator.

6.5 Experimental Results

This Section presents the vulnerable instructions, along with their relative vulnerabilities, obtained from the proposed machine learning framework—*SmartLearn*.

6.5.1 Feature Selection

Determining the set of most vulnerable instructions can be categorized as a classification problem. For each training instance, a weighted-average of the *number of bit-flips* is calculated that are induced by various droop levels during an instruction execution. A higher weight is assigned to the low-droop induced bit-flips, to signify a more vulnerable opcode. If the weighted-average is more than a preset threshold value, the instruction is inferred to be vulnerable. The *feature_importances_* method of the Random Forest classifier (in scikit-learn [110]) is used to abandon *the number of operands* as a feature due to a very poor importance. Finally, the *opcode type*, *first operand width* and *second operand width* are used as the features for training. The training set comprises 100,000 instances.

Classifier	**Average Accuracy**
Random Forest	85.5%
Support Vector Machine	85.4%
Naive Bayes	85.1%
XGBoost	85.5%

Table 6.1: Average accuracies from 10-fold cross validation.

6.5.2 Ranking Vulnerable Instructions

Table 6.1 reports the average accuracies of various classifiers from 10-fold cross validation, using the scikit-learn Python package [110]. The average accuracies for other *folds* (e.g., 5 to 9) vary within 1% compared to those reported in Table 6.1. Subsequently, the opcodes are ranked based on their inferred vulnerabilities, using equation 3 in [111]. 7 opcodes are considered in the experiments. In the decreasing order of vulnerability, the rank of the 7 opcodes are: AND, MULT, LOAD, ADD, SUB, XOR and OR. It is also found that a larger operand has a greater contribution to the vulnerability of an instruction. So, it is concluded that the top ranked opcodes, combined with large operands, are favorable to launch FIFAs in a large share of chips of the same design.

CHAPTER 7
CONCLUSION

This dissertation proposes cross-layer design techniques to improve the reliability and energy-efficiency of modern multi-core systems, while exposing a critical security loophole of low-power computing. Specifically, the presented works enable a reliable on-chip communication under a high power supply noise, improve the energy-efficiency of NTC GPUs over traditional STC ones, and demonstrate a heightened security vulnerability to fault attacks while moving from STC to NTC operating conditions.

Ensuring a fault-free communication is imperative to the performance and energy-efficiency of an NoC—the de-facto on-chip communication fabric for contemporary multi-core systems. Accommodating Moore's law to house more transistors on a chip has resulted in a severe reliability concern due to the power supply noise—a leading cause of the faulty communication in NoCs. This dissertation demonstrates that existing flow-control protocols and routing algorithms are ineffective in mitigating the voltage noise in an NoC PDN. A collection of novel flow-control protocols (PAF) and an adaptive routing algorithm (PAR) is proposed, to improve the peak PSN in NoCs. The best scheme improves the regional peak PSN by $\sim$15% and the EDP by $\sim$12%, with 4.1% average performance overhead, and marginal area and power footprints.

Near-threshold computing has been touted as an energy-efficient computing paradigm. Due to the immense opportunity in exploiting the thread-level parallelism, GPUs are proposed as a viable platform for NTC in this dissertation. However, PV-induced delay variability can severely undermine the energy-efficiency benefit of NTC GPUs. This dissertation proposes SwiftGPU—an ingenious GPU design paradigm, to tackle the potential performance and energy-efficiency hazards at NTC. SwiftGPU employs SAS, that dynamically sprints the MLDs based on the dimensions of the GPGPU kernels, as well as, the MLD usage pattern during the kernel execution. Evaluated with a rigorous cross-layer methodology, SAS achieves an average of ∼15% improvement in energy-efficiency with marginal hardware overheads, over an ideal PV-free GPU, operating at STC.

While ensuring the quality-of-service under a strict power budget is a first-order design constraint for low-power devices, the glaring security vulnerabilities that arise from a low-power operation, necessitate a design-time trade-off for energy efficiency vis-á-vis offered security. This dissertation uncovers a paradigm shift in the security vulnerability of the NTC systems, in relation to timing fault attacks. Using a cross-layer methodology, it is shown that NTC systems are, by and large, more susceptible to timing faults, despite boasting a larger timing slack compared to the traditional STC systems. A novel threat model (TITAN) is proposed that can stealthily exploit the security vulnerability of NTC, to inflict an application-level damage in a low-power system. To demonstrate the efficacy of TITAN, hardware experiments are performed using off-the-shelf commodities. Based on two proposed security metrics, (viz., *minimum glitch width* and *minimum glitch magnitude*), the experiments show a 1.6X and a 2.8X deterioration, respectively, in fault resilience of low-power operation over a traditional super-threshold operation, exposing a tremendous vulnerability of the NTC systems to TITAN.

Another threat model, called FIFA, is proposed that can also exploit this extreme security vulnerability of the NTC systems to launch a large scale fault attacks in low-power mobile platforms. FIFA employs a machine learning framework, called *SmartLearn*, to identify the vulnerabilities of many NTC chips of a given design. By learning the vulnerabilities, FIFA creates software patches to inflict potent timing fault attacks in the victimized chips.

REFERENCES

[1] Y. Hoskote, S. R. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-ghz mesh interconnect for a teraflops processor,” *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[2] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal, “On-chip interconnection architecture of the tile processor,” *Micro, IEEE*, pp. 15–31, Sept.-Oct. 2007.

[3] 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.

[4] 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 *DAC*, 2012, pp. 1147–1152.

[5] 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.

[6] T. Miller, R. Thomas, X. Pan, and R. Teodorescu, “Vrsync: Characterizing and eliminating synchronization-induced voltage emergencies in many-core processors,” in *Proc. of ISCA*, 2012, pp. 249–260.

[7] X. Hu, G. Yan, Y. Hu, and X. Li, “Orchestrator: a low-cost solution to reduce voltage emergencies for multi-threaded applications,” in *Proc. of DATE*, 2013, pp. 208–213.

[8] M. S. Gupta, V. J. Reddi, G. H. Holloway, G.-Y. Wei, and D. M. Brooks, “An event-guided approach to reducing voltage noise in processors,” in *Proc. of DATE*, 2009, pp. 160–165.

[9] S. Penolazzi and A. Jantsch, “A high level power model for the nostrum noc,” in *Proc. of DSD*, 2006, pp. 673–676.

[10] N. Dahir, T. S. T. Mak, F. Xia, and A. Yakovlev, “Modeling and tools for power supply variations analysis in networks-on-chip,” *TC*, vol. 63, no. 3, pp. 679–690, 2014.

[11] ——, “Minimizing power supply noise through harmonic mappings in networks-on-chip,” in *Proc. of CODES-ISSS*, 2012, pp. 113–122.

[12] G. Michelogiannakis, J. D. Balfour, and W. J. Dally, “Elastic-buffer flow control for on-chip networks,” in *HPCA*, 2009, pp. 151–162.

[13] Y. H. Kang, T.-J. Kwon, and J. T. Draper, “Fault-tolerant flow control in on-chip networks,” in *NOCS*, 2010, pp. 79–86.

[14] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Catra-congestion aware trapezoid-based routing algorithm for on-chip networks," *Proc. of DATE*, pp. 320–325, 2012.

[15] D. Fick, A. DeOrio, G. K. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant nocs," in *Proc. of DATE*, 2009, pp. 21–26.

[16] Y.Wang, S.Roy, and N.Ranganathan, "Run-time power-gating in caches of gpus for leakage energy savings," in *Proc. of DATE*, March 2012.

[17] J.Lee, V.Sathisha, M.Schulte, K.Compton, and N.S.Kim, "Improving throughput of power-constrained gpus using dynamic voltage/frequency and core scaling," in *PACT*, Oct. 2011.

[18] K. Ma, X. Li, W. Chen, C. Zhang, and X. Wang, "Greengpu: A holistic approach to energy efficiency in gpu-cpu heterogeneous architectures," in *ICPP*. IEEE, 2012, pp. 48–57.

[19] V. Narasiman, M. Shebanow, C. J. Lee, R. Miftakhutdinov, O. Mutlu, and Y. N. Patt, "Improving gpu performance via large warps and two-level warp scheduling," in *Proc. of MICRO*. ACM, 2011, pp. 308–317.

[20] B. Wang, B. Wu, D. Li, X. Shen, W. Yu, Y. Jiao, and J. S. Vetter, "Exploring hybrid memory for gpu energy efficiency through software-hardware co-design," in *PACT*, 2013, pp. 93–102.

[21] 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.

[22] S. Hsu, A. Agarwal, M. Anders, S. Mathew, H. Kaul, F. Sheikh, and R. Krishnamurthy, "A 280mv-to-1.1v 256b reconfigurable SIMD vector permutation engine with 2-dimensional shuffle in 22nm CMOS," in *ISSCC*, 2012, pp. 178–180.

[23] S.-M. Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," *IEEE Trans. Comput.*, vol. 49, no. 9, pp. 967–970, Sep. 2000. [Online]. Available: http://dx.doi.org/10.1109/12.869328

[24] J. Balasch, B. Gierlichs, and I. Verbauwhede, "An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus," in *Proc. of the Workshop on FDTC*. IEEE Computer Society, 2011, pp. 105–114.

[25] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Proc. of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, ser. EUROCRYPT'97. Springer-Verlag, 1997, pp. 37–51.

[26] C. H. Kim, "Improved differential fault analysis on aes key schedule," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 41–50, 2012.

[27] G. Suarez-Tangil, J. E. Tapiador, F. Lombardi, and R. Di Pietro, "Alterdroid: Differential fault analysis of obfuscated smartphone malware," *IEEE Transactions on Mobile Computing*, vol. 15, no. 4, pp. 789–802, 2016.

[28] K. Sakiyama, Y. Li, M. Iwamoto, and K. Ohta, "Information-theoretic approach to optimal differential fault analysis," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 109–120, 2012.

[29] L. Zussa, J.-M. Dutertre, J. Clediere, B. Robisson, A. Tria *et al.*, "Investigation of timing constraints violation as a fault injection means," in *27th Conference on Design of Circuits and Integrated Systems (DCIS), Avignon, France*, 2012.

[30] S. S. Mukherjee, C. T. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "Measuring architectural vulnerability factors," *IEEE Micro*, vol. 23, no. 6, pp. 70–75, 2003.

[31] B. Yuce, N. F. Ghalaty, and P. Schaumont, "Tvvf: Estimating the vulnerability of hardware cryptosystems against timing violation attacks," in *IEEE International Symposium on HOST*, 2015, pp. 72–77.

[32] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*. Springer, 1999, pp. 388–397.

[33] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 286–296. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315282

[34] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *IEEE Symposium on Security and Privacy*, 2015, pp. 605–622.

[35] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 494–505, Jun. 2007. [Online]. Available: http://doi.acm.org/10.1145/1273440.1250723

[36] R. Hund, C. Willems, and T. Holz, "Practical timing side channel attacks against kernel space aslr," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 191–205.

[37] C. Luo, Y. Fei, P. Luo, S. Mukherjee, and D. Kaeli, "Side-channel power analysis of a gpu aes implementation," in *Computer Design (ICCD), 2015 33rd IEEE International Conference on*. IEEE, 2015, pp. 281–288.

[38] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *DTC*, vol. 27, no. 1, pp. 10–25, 2010.

[39] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using ic fingerprinting," in *Proc. of Security and Privacy*, 2007, pp. 296–310.

[40] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," in *Proc. IEEE*, 2014, pp. 1229–1247.

[41] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *TVLSI*, vol. 20, no. 1, pp. 112–125, 2012.

[42] S. Narasimhan, R. S. Chakraborty, D. Du, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia, "Multiple-parameter side-channel analysis: A non-invasive hardware trojan detection approach," in *Proc. of HOST*, 2010, pp. 13–18.

[43] K. Hu, A. N. Nowroz, S. Reda, and F. Koushanfar, "High-sensitivity hardware trojan detection using multimodal characterization," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '13. San Jose, CA, USA: EDA Consortium, 2013, pp. 1271–1276. [Online]. Available: http://dl.acm.org/citation.cfm?id=2485288.2485593

[44] M. A. Ameen, J. Liu, and K. Kwak, "Security and privacy issues in wireless sensor networks for healthcare applications," *J. Medical Systems*, pp. 93–101, 2012.

[45] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004, pp. 162–175.

[46] A. J. Burns, M. E. Johnson, and P. Honeyman, "A brief chronology of medical device security," vol. 59, no. 10, pp. 66–72, 2016.

[47] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, 2008, pp. 129–142.

[48] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, pp. 521–534, 2002.

[49] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. R. Vangal, G. Ruhl, and N. Borkar, "A 2 tb/s 6 , $^{\times}$, 4 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS," *J. of Solid-State Circ.*, vol. 46, no. 4, pp. 757–766, 2011.

[50] S. Ma, N. D. E. Jerger, and Z. Wang, "Dbar: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *Proc. of ISCA*, 2011, pp. 413–424.

[51] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *NOCS*, 2012, pp. 201–210.

[52] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *T. Electron Devices*, pp. 2816 –2823, 2006.

[53] N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, “A detailed and flexible cycle-accurate network-on-chip simulator,” in *ISPASS*, 2013, pp. 86–96.

[54] R. Bertran, A. Buyuktosunoglu, P. Bose, T. J. Slegel, G. Salem, S. M. Carey, R. F. Rizzolo, and T. Strach, “Voltage noise in multi-core processors: Empirical characterization and optimization opportunities,” in *Proc. of MICRO*, 2014, pp. 368–380.

[55] T. E. Carlson, W. Heirman, and L. Eeckhout, “Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation,” in *Proc. of SC*, 2011.

[56] B. Kim and V. Stojanovic, “Characterization of equalized and repeated interconnects for noc applications,” *IEEE Design & Test of Computers*, vol. 25, no. 5, pp. 430–439, 2008.

[57] “International technology roadmap for semiconductors, 2013 edition, http://www.itrs.net.”

[58] P. Gratz, B. Grot, and S. W. Keckler, “Regional congestion awareness for load balance in networks-on-chip,” in *HPCA*, 2008, pp. 203–214.

[59] L.-R. Zheng and H. Tenhunen, “Fast modeling of core switching noise on distributed lrc power grid in ulsi circuits,” *IEEE transactions on advanced packaging*, vol. 24, no. 3, pp. 245–254, 2001.

[60] D. Becker. (2012, August) Open source noc router rtl. [Online]. Available: https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router

[61] J. Lucas, S. Lal, M. Andersch, M. Alvarez-Mesa, and B. Juurlink, “How a single chip causes massive power bills gpusimpow: A gpgpu power simulator,” in *ISPASS*, April 2013.

[62] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, “Debunking the 100x GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU,” in *Proc. of ISCA*, 2010, pp. 451–460.

[63] L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, R. H. Dennard, and W. Haensch, “Practical strategies for power-efficient computing technologies,” *Proc. IEEE*, vol. 98, no. 2, pp. 215–236, 2010.

[64] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, “ Multi2Sim: A Simulation Framework for CPU-GPU Computing ,” in *PACT*, Sep. 2012.

[65] “ AMD Accelerated Parallel Processing (APP) Software Development Kit ,” 2016. [Online]. Available: http://developer.amd.com/sdks/amdappsdk/

[66] C. Silvano, G. Palermo, S. Xydis, and I. S. Stamelakos, “Voltage island management in near threshold manycore architectures to mitigate dark silicon,” in *Proc. of DATE*, 2014, pp. 1–6.

[67] Y. Pu, X. Zhang, J. Huang, A. Muramatsu, M. Nomura, K. Hirairi, H. Takata, T. Sakurabayashi, S. Miyano, M. Takamiya, and T. Sakurai, "Misleading energy and performance claims in sub/near threshold digital systems," in *Proc. of ICCAD*, 2010, pp. 625–631.

[68] 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.

[69] T. N. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu, "Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips," in *HPCA*, 2012, pp. 1–12.

[70] X. He, G. Yan, Y. Han, and X. Li, "Superrange: Wide operational range power delivery design for both stv and ntv computing," in *DATE*, 2014, pp. 1–6.

[71] S. P. Mohanty and D. K. Pradhan, "ULS: A dual-V_{th}/high-kappa nano-cmos universal level shifter for system-level power management," *J Emerg. Tech. Comp. Sys.*, vol. 6, no. 2, 2010.

[72] J. Leng, T. H. Hetherington, A. ElTantawy, S. Z. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "Gpuwattch: enabling energy optimizations in gpgpus," in *Proc. of ISCA*, 2013, pp. 487–498.

[73] W. Zhao and Y. Cao, "Predictive technology model," June 2012. [Online]. Available: http://ptm.asu.edu/

[74] B. Calhoun and A. Chandrakasan, "A 256-kb 65-nm sub-threshold sram design for ultra-low-voltage operation," in *J. of Solid-State Circ.*, vol. 42, no. 3, March 2007, pp. 680–688.

[75] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[76] N. Magen, A. Kolodny, U. C. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor," in *Sys. Level Intercon. Pred.*, 2004, pp. 7–13.

[77] *MIAOW GPU*, http://miaowgpu.org.

[78] E. Krimer, P. Chiang, and M. Erez, "Lane decoupling for improving the timing-error resiliency of wide-simd architectures," in *Proc. of ISCA*, 2012, pp. 237–248.

[79] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health monitoring and management using internet-of-things (iot) sensing with cloud-based processing: Opportunities and challenges," in *2015 IEEE International Conference on Services Computing, SCC 2015, New York City, NY, USA, June 27 - July 2, 2015*, 2015, pp. 285–292.

[80] S. Seo, R. G. Dreslinski, M. Woh, Y. Park, C. Chakrabarti, S. A. Mahlke, D. Blaauw, and T. N. Mudge, "Process variation in near-threshold wide simd architectures," in *Proc. of DAC*, 2012, pp. 980–987.

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

[82] 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.

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

[84] 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.

[85] Y. Kim, L. K. John, S. Pant, S. Manne, M. J. Schulte, W. L. Bircher, and M. S. S. Govindan, “Audit: Stress testing the automatic way,” in *Proc. of MICRO*, 2012, pp. 212–223.

[86] V. Southmayd, “4 patch management practices to keep your clients secure,” August 2016, http://www.labtechsoftware.com/blog/4-patch-management-best-practices/.

[87] K. Dunham, *Mobile malware attacks and defense.* Syngress, 2008.

[88] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. X. Song, and H. Yin, “Automatically identifying trigger-based behavior in malware,” in *Botnet Detection: Countering the Largest Security Threat*, 2008, pp. 65–88.

[89] X. Jiang and Y. Zhou, *A Survey of Android Malware.* New York, NY: Springer New York, 2013, pp. 3–20.

[90] R. Bertran, A. Buyuktosunoglu, P. Bose, T. J. Slegel, G. Salem, S. M. Carey, R. F. Rizzolo, and T. Strach, “Voltage noise in multi-core processors: Empirical characterization and optimization opportunities,” in *Proc. of MICRO*, 2014, pp. 368–380.

[91] X. Zhang, T. Tong, S. Kanev, S. K. Lee, G. Wei, and D. M. Brooks, “Characterizing and evaluating voltage noise in multi-core near-threshold processors,” in *ISLPED*, 2013, pp. 82–87.

[92] N. Ahmed, M. Tehranipoor, and V. Jayaram, “Supply voltage noise aware atpg for transition delay faults,” in *Proc. of VTS*, 2007, pp. 179–186.

[93] A. Todri and M. Marek-Sadowska, “Power delivery for multicore systems,” *TVLSI*, vol. 19, no. 12, pp. 2243–2255, 2011.

[94] W. Lee, Y. Wang, and M. Pedram, “Optimizing a reconfigurable power distribution network in a multicore platform,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1110–1123, 2015.

[95] Setting a process’s cpu affinity mask. https://linux.die.net/man/2/sched_setaffinity.

[96] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Trans. Systems, Man, and Cybernetics, Part C*, vol. 40, no. 1, pp. 1–12, 2010.

[97] *Claremont: Intel Near-Threshold Voltage Processor*, http://www.realworldtech.com/near-threshold-voltage/.

[98] *Raspberry-Pi Official Documentation*, https://www.raspberrypi.org/documentation/configuration/config-txt/overclocking.md.

[99] A. Barenghi, G. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi, "Low voltage fault attacks to AES and RSA on general purpose processors," *IACR Cryptology ePrint Archive*, vol. 2010, p. 130, 2010.

[100] *Raspberry-Pi Schematics*, https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/README.md.

[101] Y. Kim and L. K. John, "Automated di/dt stressmark generation for microprocessor power delivery networks," in *ISLPED*, 2011, pp. 253–258.

[102] J. Constantin, A. P. Burg, Z. Wang, A. Chattopadhyay, and G. Karakonstantis, "Statistical fault injection for impact-evaluation of timing errors on application performance," in *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*, 2016, pp. 13:1–13:6.

[103] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.

[104] E. El-Qawasmeh and K. Al-Noubani, "A polynomial time algorithm for the n-queens problem," in *Proc. of the IASTED on Neural Networks and Computational Intelligence*, 2004, pp. 191–195.

[105] 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.

[106] T. Wei, C. Mao, A. B. Jeng, H. Lee, H. Wang, and D. Wu, "Android malware detection via a latent network behavior analysis," in *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012, Liverpool, United Kingdom, 2012*, 2012, pp. 1251–1258.

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

[108] A. M. Joshi, L. Eeckhout, L. K. John, and C. Isen, "Automated microprocessor stressmark generation," in *14th International Conference on High-Performance Computer Architecture (HPCA-14 2008), 16-20 February 2008, Salt Lake City, UT, USA*, 2008, pp. 229–239.

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

[110] scikit-learn, machine learning in python. https://scikit-learn.org.

[111] F. Pagnotta and H. Amran, “Using data mining to predict secondary school student alcohol consumption,” *Department of Computer Science, University of Camerino*, 2016.

CURRICULUM VITAE

Prabal Basu

Published Journal Articles

- 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 (TETC)*, 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 (ESL)*, vol. 10, issue. 4, pp. 115-118, 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, issue 3, pp. 49, 2017.

- IcoNoClast: Tackling Voltage Noise in the NoC Power Supply Through Flow-Control and Routing Algorithms. Prabal Basu, Rajesh Jayashankara Shridevi, Koushik Chakraborty and Sanghamitra Roy. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 25, issue 7, pp. 2035-2044, 2017.

Published Conference Papers

- GreenTPU: Improving Timing Error Resilience of a Near-Threshold Tensor Processing Unit. Pramesh Pandey, Prabal Basu, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM Design Automation Conference (DAC)*, 2019.

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

- 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. *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2018.

- SwiftGPU: Fostering Energy Efficiency in a Near-Threshold GPU Through a Tactical Performance Boost. Prabal Basu, Hu Chen, Shamik Saha, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM Design Automation Conference (DAC)*, 2016.

- PRADA: Combating Voltage Noise in the NoC Power Supply Through Flow-Control and Routing Algorithms. Prabal Basu, Rajesh JS, Koushik Chakraborty and Sanghamitra Roy. *IEEE/ACM Design Automation and Test in Europe (DATE)*, 2016.