# Exploiting Adaptive Techniques to Improve Processor Energy Efficiency

Hu Chen

*Utah State University*

# EXPLOITING ADAPTIVE TECHNIQUES TO IMPROVE PROCESSOR ENERGY EFFICIENCY

by

Hu Chen

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

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

UTAH STATE UNIVERSITY
Logan, Utah

2016

## ABSTRACT

Exploiting Adaptive Techniques to Improve Processor Energy Efficiency

by

Hu Chen, Doctor of Philosophy

Utah State University, 2016

Major Professor: Sanghamitra Roy, PhD
Department: Electrical and Computer Engineering

Rapid device-miniaturization keeps on inducing challenges in building energy efficient microprocessors. As the size of the transistors continuously decreasing, more uncertainties emerge in their operations. On the other hand, integrating more and more transistors on a single chip accentuates the need to lower its supply-voltage. This dissertation investigates one of the primary device uncertainties — timing error, in microprocessor pipeline; as well as, the energy trade-off between processor components in Near-Threshold Computing (NTC) era. Using rigorous cross-layer methodology, this dissertation identifies novel opportunities lying in microprocessor workload, and the shifted processor performance bottleneck in NTC era. Then it proposes various innovative techniques to exploit these opportunities to maintain processor energy efficiency, in the context of emerging challenges. Evaluated with the cross-layer methodology, the proposed approaches achieve substantial improvements in processor energy efficiency, compared to other start-of-art techniques.

(91 pages)

## PUBLIC ABSTRACT

Exploiting Adaptive Techniques to Improve Processor Energy Efficiency

by

Hu Chen, Doctor of Philosophy

Utah State University, 2016

Major Professor: Sanghamitra Roy, PhD
Department: Electrical and Computer Engineering

As technology advances, the size of transistor — the basic building block of microprocessors, continues shrinking. The downscaled transistor will make its manufacture process become less controllable, and its operation less predictable. Moreover, smaller transistor size will lead to higher volume of transistors on a single chip, thus increases the chip power consumption. This dissertation proposes various techniques to efficiently tolerate the uncertainties in microprocessors. Also, it proposes an innovative design paradigm to build energy-efficient processor under ultra-low supply-voltage. Evaluated with rigorous methodology, the proposed approaches achieve significant improvement in processor energy efficiency, compared to other state-of-art techniques.

*To my wife, Jing and our adorable son Jacob*

## ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Sanghamitra Roy, and my co-advisor Dr. Koushik Chakraborty for the continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. Without it, my whole Ph.D study and research would not be never become possible.

I would like to thank all of my supervisory committee: Dr. Rose Hu, Dr. Ryan Gerdes, Dr. Haitao Wang, and Dr. Ming Li, for their insightful comments and encouragement that have incented me to widen my research from various perspectives.

My sincere thanks also goes to my fellow labmates in USU Bridge Lab, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years. I would like to thank Yiding, Jason and Dean for their guidance as I started my research; thank Manzi, Shamik, Prabal and Atif for withstanding the deadline pressure with me; thank to Shayan, Rajesh and Harshita for all the fun we had in the 2014 DAC trip; thank to Saptarshi for the fun conversations we had about life; thank to all the students who have helped me in one way or another: Brennan, Mohammed, McCabe, Brian, Andrew, Kenneth, Michael, Kenyon, Kurt and Chidham.

I would like to thank the ECE Department and all of the staff members, for giving me the opportunity to pursue my PhD degree. I am very thankful to Mary Lee Anderson, Kathy Phippen and Tricia Brandenburg for the invaluable support in supporting my Ph.D study in many ways.

Last but not the least, I would like to thank my family: my wife Jing and my son Jacob, for being my everlasting source of joy and motivation; my father for his impact on my life; my mother for her unconditional love.

Hu Chen

# CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

The utilization of computers is one of the key attributes that define modern society. Processors are the core components of a computer system. With rapid miniaturizations of semiconductor devices, a single processor contains more and more transistors. This device miniaturization introduces complicated reliability and energy-efficiency hazards in circuits' operation. First of all, the downscaled device size increases the device sensitivities towards both process variations, as well as operation environment. These sensitivities create various uncertainties in device operation, thereby affect the performance and reliability of the whole system. On the other hand, the fast-increasing transistor count leads to a upsurge in the power consumption of a single chip. As computers become ubiquitous in the daily life of human beings, the power consumption of the processors become one of the critical concerns in building our energy-efficient society.

Among all the reliability challenges, the uncertainty in device delay is of particular concern. The device delay uncertainty can potentially manifest as timing errors [7, 50, 61], thereby cause malfunction in the whole system. Previous works have intensively studied various sources of the delay variations, including Process Variation (PV) and aging [50, 61]. Different from these well-known sources, this work reveals that variation in delays of pipeline stages also heavily depends on specific applications running on the microprocessor. For instance, sensitized paths during the execution of real world applications may exhibit strikingly distinct characteristics than expected from a purely static timing analysis. Observing this critical source of delay variation, this work combines early error prediction with clock skew tuning to propose *Dynamically Adaptable Resilient Pipeline* (DARP), which outlines a next wave of innovation in pushing the energy efficient frontier of pipelined microprocessor design.

Supply voltage (Vdd) is one of the key factors that decide the energy efficiency of the Integrated Circuits (IC). Reducing Vdd will effectively decrease the power dissipation of the transistor. Recently, Near-Threshold Computing (NTC), where Vdd is marginally higher than transistor threshold voltage (Vth), has emerged as a promising direction to improve the energy-efficiency of integrated circuits. However, such a reduction in Vdd will significantly increase the transistor delay and its sensitivity to PV. Therefore, the NTC processor cores will operate on a much degraded frequency compared to Super-Threshold Computing (STC) processor cores [18, 39, 53]. Also, the achievable operating frequency of each core can vary substantially in a NTC multi/many-core system [18, 39, 53]. Aside from these well-known challenges, this work reveals another significant change as migrating the processors into NTC era: the historic performance gap between core and memory will be largely superseded by the performance bottleneck within the core. Based on this observation, this work proposes *Turbo Execution*, which exploits this new performance knob in processor cores to compensate the severe performance degradation and variability in NTC era.

## 1.1 Contribution of This Dissertation

The research works associating with this dissertation have been partially published in various conference and journals, including *2016 IEEE/ACM Design Automation Conference (DAC)*, *2015 IEEE/ACM Design Automation Conference (DAC)*, *2014 IEEE/ACM Design, Automation and Test in Europe (DATE)*, *2014 IEEE International Symposium on Quality Electronic Design (ISQED)*, and *2015 ACM Transactions on Design Automation of Electronic Systems (TODAES).*

Publications stemming from this dissertation are listed as follows:

### 1.1.1 Conference Papers

- Opportunistic turbo execution in NTC: exploiting the paradigm shift in performance bottlenecks. Hu Chen, Dieudonne Manzi, Sanghamitra Roy, Koushik Chakraborty. *2015 IEEE/ACM Design Automation Conference (DAC).*

- DARP: Dynamically Adaptable Resilient Pipeline Design in Microprocessors. Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2014 IEEE/ACM Design, Automation and Test in Europe (DATE).*

- Exploiting Static and Dynamic Locality of Timing Errors in Robust L1 Cache Design. Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2014 IEEE International Symposium on Quality Electronic Design (ISQED).*

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

- Synergistic Timing Speculation for Multi-threaded Programs. Atif Yasin, Jeff Zhang, Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2016 IEEE/ACM Design Automation Conference (DAC).*

### 1.1.2 Journal Paper

- DARP-MP: Dynamically Adaptable Resilient Pipeline Design in Multicore Processors. Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2015 ACM Transactions on Design Automation of Electronic Systems (TODAES).*

# CHAPTER 2

# LITERATURE REVIEW

This chapter gives a comprehensive literature survey on recent works that associated with processor energy-efficiency in various aspects. Chapter 2.1 lists and explains the state-of-the-art recent works. Chapter 2.2 positions the contributions of this dissertation on efficiently tolerating timing errors in microprocessor components. Chapter 2.3 clearly delineates the position of this research on designing energy-efficient multi/many-core systems in NTC era.

## 2.1 State-of-the-Art Works in Processor Reliability and Energy-Efficiency

The existing works that most relevant to this dissertation can be broadly classified into five categories: basic analysis and modeling of the device reliability (Section 2.1.1); circuit techniques to efficiently cope with timing uncertainties in devices (Section 2.1.2); architectural strategies to efficiently tolerate device timing variations (Section 2.1.3); the fundamental opportunities and challenges in Near-Threshold Computing (NTC) era (Section 2.1.4); the state-of-the-art design techniques in NTC (Section 2.1.5).

### 2.1.1 Analysis and Modeling of Device Reliability

- **Borkar [7]**: Borkar presents technology and economic challenges in 22nm node and beyond. As technology continues scaling down, the static and dynamic variations in devices will continue to become worse. Therefore, one of the major design challenges in lower technology node is: how to build a reliable yet energy-efficient system, out of unreliable components.

- **IVF [50]**: IVF characterizes the vulnerability of microprocessor structures to various intermittent faults. The timing-faults, where the device delay violates the design constraints, are mainly caused by inductive noises, aging, crosstalk, or process, voltage, temperature (PVT) variations. They usually lead to write wrong data to storage cells and finally become reliability problem.

- **VARIUS [61]**: VARIUS models the impact of within-die parameter variation upon the device delay. Using a small number of highly intuitive parameters, VARIUS counts in both random and systematic effect of parameter variation. Also, VARIUS proposes a framework to model timing errors in microarchitectures that caused by parameter variation. The model gives the failure rate of a microarchitecture as a function of clock frequency and the amount of variation.

- **Borkar [64]**: Borkar presents the sources of variability in devices as well as the current variation-tolerant techniques that have been employed. Random dopant fluctuation, sub-wavelength lithography, as well as various dynamic fluctuations during the device operation, are the major sources of the device variability. He also predicts the device variation to increase as the technology continues scaling further, and proposes several potential solutions to address these reliability challenges.

- **Mukhopadhyay et al. [45]**: Mukhopadhyay et al. analyze the failure probabilities (access-time failure, read/write failure and hold failure) in SRAM, as well as, develop a method to predict the yield of a memory chip. They also propose a statistical design strategy for the SRAM cell and the memory, to minimize the failure probability of a memory chip under area and leakage constraints. The developed design strategy statistically sizes different transistors of the SRAM cell and optimizes the number of redundant columns to be used in the SRAM array.

- **Agarwal et al. [5]**: Agarwal et al. analyze various SRAM cell failures under Process Variation (PV) and propose a new PV-tolerant cache architecture suitable for high-performance applications. The proposed technique dynamically tracks the faulty cells under the given operation condition, and employ address-remapping to avoid accessing the faulty cells. This scheme is transparent to processor architecture and has negligible energy and area overhead.

### 2.1.2 Efficiently Coping with Device Timing Uncertainties at Circuit Level

- **Razor** [**17**]: Razor proposes the timing-speculation technique to remove the conventional timing guardband in circuits. Instead of using enough timing guardband to ensure always timing-error-free operation, it employs double-sampling infrastructure to detect occurrence of timing-error, then replay the erroneous instruction to resume correct execution. Under infrequent error occurrence, this technique can effectively improve both the performance and the energy-efficiency of the microprocessor.
- **Bowman et al.** [**8**]: Bowman et al. implemented novel timing-error detection and recovery circuits to eliminate the timing guardband. The proposed microarchitecture lowers the clock energy, as well as, simplifies the management of metastability, in the Error-Detection Sequential (EDS) circuits. On the other hand, the error-recovery circuits replay the erroneous instructions at lower clock frequency to ensure correct functionality.
- **TIMBER** [**16**]: TIMBER masks timing errors online by enabling timing-borrowing between microprocessor pipeline stages. Like timing-speculation techniques, TIMBER employs double-sampling infrastructure to detect timing-error occurrence. However, instead of instruction replay or roll-back, TIMBER seamlessly redistributes timing budget across consecutive pipeline stages to mask the timing errors on the fly. Two state-of-art structures: TIMBER flip-flop and TIMBER latch are proposed to enable the timing-borrowing.
- **Ghazemazar et al.** [**19**]: Ghazemazar et al. leverage timing-borrowing technique to optimize the energy-efficiency of a synchronous linear pipeline circuit. Employing Soft-Edge Flip-Flops (SEFF), the proposed approach enables time borrowing between pipeline stages to provide the timing-critical stages with more time to complete their

operation. With the timing-borrowing infrastructure, the proposed technique also dynamically adjusts the supply voltage level and clock frequency of the pipeline, to achieve the best throughput-energy trade-off.

- **Bowman et al. [9]**: Bowman et al. propose various dynamic-adjusting techniques to tolerant physical condition variations in the circuits. Sensors and adaptive voltage and frequency circuits work together to provide proper configuration for the circuit. On the other hand, canary-circuits, as well as error detection and error recovery circuits are employed to efficiently avoid/tolerate timing errors in microprocessors. The authors discuss the opportunity of incorporating these techniques in Computer-aided Design (CAD) tools.

- **AutoRex [69]**: AutoRex proposes an automatic tool to do post-silicon clock-tuning, therefore improves the nominal frequency of the processor. Although commonly available in high-end microprocessors, the huge amount of possible configurations for the clock-tuning elements poses a significant challenge for its optimal usage. Also, Process Variation (PV) can make the optimal configuration vary for different chips. AutoRex operates by taking data from a volume experiment across multiple process corners and then using Satisfiability Modulo Theory (SMT) solver to creates optimal assignments for the tunable delay buffers in each chip.

- **Ye et al. [75]**: Ye et al. employs both offline and online clock-skewing technique to tolerate PV in microarchitecture. By investigating the timing-criticality of neighbor stages, offline clock-skewing technique can statically borrow time into timing-critical stages to eliminate the timing error occurrence. On the other hand, the timing budget for back-to-back timing-critical stages are dynamically adjusted online to minimize the overall timing-error occurrence. Razor-like timing-speculation technique is employed to protect the circuits as timing error occurs.

- **Lak et al. [31]**: Lak et al. uses clock-skew tuning to address the lifetime performance degradation caused by device aging. Their technique employs in-situ sensor to detect the aging-induced degradation in path delay as it approaches the limit of clock cycle-time. Then the clock skew is adjusted for that path to give it enough timing guardband. At the same time, an optimization algorithm is launched to adjust the clock-tuning elements across the whole circuit to maintain a highest achievable clock frequency.

- **Pan et al. [51]**: Pan et al. present an analysis of a representative high-performance processor architecture and show that the caches have the highest probability of causing yield losses under PV. Then they propose voltage-boosting to eliminate the PV-induced timing-errors in cache. The wordlines with timing-errors are located during the test, then the voltage of these wordlines are statically boosted to reduce the latency.

- **Gregg et al. [20]**: Gregg et al. employs Adaptive-Body-Biasing (ABB) to mitigate the PV-induced delay degradation in cache accesses. Using two algorithms to find near-optimal configurations, the body biases are generated on set-granularity to compensate the PV-induced cache access timing degradation.

### 2.1.3 Architectural Strategies to Efficiently Tolerate Device Timing Variations

- **Xin et al. [74]**: Xin et al. demonstrate the pronounced locality in instruction-level timing-error occurrence in microprocessor pipeline. With this observance, they propose a timing error prediction technique, which uses Program Counter (PC) to dynamically anticipate timing errors at instruction-level; as well as, an error padding technique which stalls the pipeline for one-cycle to cover the potential timing-error occurrence, therefore avoid the full recovery cost of timing errors.

- **Roy et al. [60]**: Roy et al. reveal that the timing error occurrence is not only associated with the instruction instance, but also related to the context of that instruction, as well as the thermal and voltage condition of the processor. Based on this discovery, they propose an effective timing-error prediction technique for the microprocessor pipeline. On a timing-error occurrence, the Program Counter (PC) of the instruction, the Branch History Register (BHR), as well as the pipe stage where the timing-error occurred, are recorded in the Timing Violation Predictor (TVP). On-chip sensors are employed to increase the prediction accuracy.

- **Chakraborty et al.** [**11**]: Chakraborty et al. improve the energy-efficiency of timing-error prediction in microprocessor pipeline. To support efficient error-padding after an timing-error being predicted, they explore various optimization techniques in the instruction scheduling in an Out-of-Order pipeline. With the proposed techniques, only the predicted-to-be- erroneous pipeline stage will stall for one-cycle to cover the potential timing error, rather than stalling the whole pipeline, as employed in existing timing- error prediction techniques.

- **Rahimi et al.** [**57**]: Rahimi et al. propose adaptive timing guardbanding to protect applications from Process-Voltage-Temperature (PVT) induced CMOS variability. Besides the intrinsic requirement of the application, the author use Instruction-Level Vulnerability (ILV) and Sequence-Level Vulnerability (SLV) to characterize the application's vulnerability to transistor timing variation. On the other hand, low-overhead Critical Path Monitors (CPM) are employed to detect the transistor timing degradation. Based on the application metadata, as well as the transistor operation status, the cycle time of the processor is dynamically adjusted to eliminate the unnecessary guardbands.

- **AVICA** [**22**]: AVICA pads an extra cycle in every L1 cache access to avoid PV-induced timing-errors. A pseudo multi-bank technique is employed to allow two consecutive cache accesses to overlap each other, thereby minimizing the performance impact of increased cache access latency. Meanwhile, several other architectural techniques are also employed to reduce the chance of bank-conflicting, where two consecutive cache accesses are targeting at the same bank thus cannot overlap each other.

- **Ozdemir et al.** [**49**]: Ozdemir et al. propose several techniques to tackle the PV-induced fault in cache. The cache ways/ lines suffering from high leakage and delay violation will be power-off to save the power consumption, as well as maintain the performance of the cache. On the other hand, they design a VAriable-latency Cache Architecture (VACA) to allow load accesses to complete with varying latencies, thus tolerating the PV-induced delay-increase in cache access. An combination of these techniques is also explored to further improve the yield of the cache memory.

- **Mutyam et al.** [**47**]: Mutyam et al. propose a scheme to tolerate the PV-induced latency degradation in SRAM cell access. During chip test, the timing-error sites of the cache are detected, and their locations are recorded in a table. At runtime an early predicted address is used to access the table, thus the timing error prediction can be made before doing cache-access. An extra cycle is padded in cache access, as an timing error has been predicted. While pipeline flush and instruction replay are initiated when there is a false negative in the prediction.

- **Kim et al.** [**28**]: Kim et al. design a 2-D error- coding to correct multi-bit errors in embedded memories. Conventional 1-D ECC schemes can neither detect nor correct large-scale multi-bit errors without incurring large performance, area and power overhead. Kim et al., on the other hand, integrate vertical error coding across words in addition to conventional per-word horizontal error coding, thereby more efficiently protect embedded memories from multi-bits errors.

- **TM** [**59**]: Instead of implementing highly costly Dynamic Voltage Frequency Scaling (DVFS) in multicore processors, Thread-Motion (TM) employs static VFS on each core, while enabling thread migration to optimize the core utilization. Different from rapidly adjusting the voltage and frequency configuration (as done in DVFS) in each core to meet the temporally varying computing needs of the running applications, TM transfers the temporary application data among the cores, then restore application execution after the transfer. An optimization algorithm is employed to search for an efficient task-distribution on the cores.

- **THPH et al. [12]**: THPH is proposed as superior alternative to conventional VFS technique. Conventional VFS adjusts the core voltage and frequency according to the computing needs of the running applications, however, it cannot alter the intrinsic characteristic of the circuits. On the other hand, THPH design a core from ground-up for a specified voltage-frequency point. As operating on that voltage-frequency point, such a core is substantially more energy-efficient than a core that is designed for other voltage-frequency but using VFS to work on that point.

### 2.1.4 Fundamental Principles in NTC Era

- **Dreslinski et al. [18]**: Dreslinski et al. systematically introduce the opportunities and challenges–as well as their possible solutions–associated with Near-Threshold Computing (NTC) era. By reducing the supply voltage to be just marginally higher than the transistor threshold voltage, NTC can significantly save the circuit energy consumption. NTC is particularly attractive in energy-critical platforms, such as high- performance date-center, mobile computing and sensors. However, performance degradation/variation and functional failure are the major challenges as migrating the circuits into NTC era. The authors review various techniques on architecture/microarchitecture level, as well as circuit level as the solution to these challenges.

- **Marković et al. [39]**: Marković et al. demonstrate a new design philosophy for the NTC era. According to their work, gate sizing–the previously widely used trade-off knob in Super-Threshold Computing (STC), will be much less effective in NTC region. On the other hand, the delay- voltage sensitivity will be much larger in NTC than that in STC. Therefore, voltage will be the most effective parameter to adjust the device delay, in NTC region. Based on this discovery, the authors propose multi-threshold(Vth) design on the chip to improve the energy efficiency in NTC era.

- **Pinckney et al. [53]**: Pinckney et al. assess the theoretical limit of energy-efficiency that can be achieved by voltage downscaling. Conventionally, parallel computing will be employed to compensate the performance degradation from voltage downscaling. However, Pinckney et al. reveal that various major energy overheads, including device leakage energy, Amdahl overhead, as well as architectural overhead, will prevent the supply voltage from continuous downscaling. According their work, the NTC region, which locates slightly above the minimal-energy point, will have leakage comprise a significant part of the total energy consumption.

- **Karpuzcu et al. [26]**: Karpuzcu et al. investigate the impact of PV upon device performance in NTC, as well as the strategies to address this challenge. The timing guardband strategy in Super-Threshold Computing (STC) will be inappropriate to apply in NTC, where the device speed is substantially slow. Also, supply voltage and threshold voltage adjusting techniques will be much limited, due to the wide core-wise variation and and the large core volume in a NTC processor. Instead, the author propose multi-frequency domain on a NTC processor to cope with PV-induced device timing variation.

### 2.1.5 NTC Design Techniques

- **Seok et al. [66]**: Seok et al. propose super-pipelining to reduce the energy consumption in NTC region. As leakage being the dominant part of the device energy consumption in NTC era, the authors employs a deeper pipeline to increase circuit's activity factor and reduce the idle time of the devices. Therefore, the circuits moves back into the dynamic-energy dominating region, and the energy wasted in leakage is reduced. Also, the authors propose novel microarchitectures to improve the circuits reliability in NTC region.

- **Wang et al. [71]**: Wang et al. explore platform energy-reduction in NTC region. Different than previous works that only consider the energy efficiency within the processor, they consider the energy-consumption in both voltage-regulators and off-chip memory accesses when optimize the platform energy. As larger chip area will decrease the manufacture yield, they also explore the trade-off between platform energy and chip area. The major design space they explored is the size of the on-chip L2 cache. As the performance gap between processor and memory shrinks in NTC, incorporating large L2 cache will be less effective in boosting processor performance.

- **Hsu et al. [24]**: Hsu et al. propose a reconfigurable SIMD vector permutation engine that can work on near-threshold supply-voltage. Various microarchitectural and circuit techniques are employed to tolerate severe process variation in NTC region.

- **Tokunaga et al. [70]**: Tokunaga et al. implement NTC graphics execution core on Intel's 22nm technology node. To maintain noise margin in memory cells while the logic cells can operate on lower supply voltage, the authors employs a dual-Vdd technique. Moreover, an adaptive clocking technique is implemented to proactively gate or divide the core clock, when high- frequency voltage droops are detected. Also, the authors propose a state-retentive sleeping mode in the Graphics Register File (GRF) to reduce the leakage energy, which is the major source of circuit energy consumption in NTC.

- **EnergySmart [27]**: EnergySmart proposes a multi-frequency domain design paradigm to cope with PV-induced performance heterogeneity in NTC multi/many-core system. Severe sensitivity to PV creates wide performance variation across cores on a NTC chip. However, the large core volume on the NTC chip makes it difficult to customize a voltage domain for each core. Instead, EnergySmart only enables multiple frequency domains on the NTC chip, while providing energy-efficiency aware job assignments to find an appropriate set of core for each job.

- **Liu et al. [35]**: Liu et al. implement an asynchronous neural signal processor in NTC. The intrinsic characteristic of asynchronous circuit makes it an attractive candidate to address two major challenges in NTC era: the severe sensitivity to PV and the dominating leakage energy consumption. On the other hand, biomedical sensor is one of the emerging applications that demand for ultra-low power and energy performance.

- **Moon et al. [44]**: Moon et al. realize a Phase-Locked Loop that work in NTC region, on 65nm technology node. Multiple circuit techniques are integrated in body-biasing of Voltage Controlled Oscillator (VCO) to mitigate severe Process-Voltage-Temperature (PVT) effect in NTC. Also, the PLL employs a novel ultra-low voltage charge pump that compensates current mismatch with an active loop filter.

- **Zhang et al. [76]**: Zhang et al. propose a receiver that operates from a single voltage supply of 300mV, therefore can be directly powered from various energy harvesting sources. Forward-Body-Biasing (FBB) is employed to temporarily boost the device performance. The proposed design also extensively utilizes transformer coupling between stages to reduce headroom requirements.

- **Hsieh et al. [23]**: Hsieh et al. propose an all digital push-pull linear voltage regulator that can operate for both STC and NTC region. A response time constraint is developed to provide a design guideline for the digital control system. It describes the correlation between required speed of the digital control system, the output performance and the size of the decoupling capacitor. A time interleaving control technique is then proposed to have a trade-off between output performance, quiescent current, and the size of decoupling capacitor.

- **Kutila et al. [30]**: Rather than exploring other alternative SRAM cells (for example, 8T, 10T, et cetera), Kutila et al. investigate the opportunity of conventional 6T SRAM cell in NTC region, targeting at low performance applications. According to their work, the two inner NMOS transistors in the 6T SRAM cell are the crux in NTC operation. By doubling their widths from the minimum, the reliability and the leakage energy consumption of the SRAM cell are substantially improved.

- **Xie et al. [73]**: Xie et al. explore the energy efficiency of Soft-Edge Flip-Flop (SEFF) based pipeline, for both STC and NTC regions. SEFF allows opportunistic timing borrowing between pipeline stages, however, applying this technique to NTC region faces a significant challenges due to large circuit parameter variations that result from manufacturing process imperfections and substrate temperature changes. The authors propose a novel delay line structure for the SEFF to provide appropriate transparency window control for the NTC region. Then a design optimization algorithm is employed to achieve the minimal Energy-Delay-Product (EDP) for the SEFF based pipeline.

- **Lee et al. [32]**: Lee et al. evaluate the voltage stacking technique in the context of NTC multicore processor. By stacking logic blocks on top of each other, voltage stacking reduces the chip leakage energy and simplifies off-chip power delivery. Within-die voltage noise tends to increase due to inter-layer current mismatch. However, unlike conventional power delivery schemes, supply rail impedance in voltage stacked systems depend on aggregated power consumption, leading to better noise immunity for high power (low impedance) operation for many-core processors.
- **Booster [42]**: Booster is a two power supply rails framework that can dynamically re-balance severe PV-induced performance heterogeneity under ultra-low supply voltage. Each core can be dynamically assigned to either of the two rails using a gating circuit. This allows cores to quickly switch between two different frequencies. Subject to the power constraint, an on-chip governor controls the timing of the switching and the time spent on each rail. Thereby, not only the PV-induced performance heterogeneity, but also the imbalance in workloads can be rebalanced by Booster.

## 2.2 Tackling Timing-Error in Microprocessors

Previous works most relevant to tolerating timing-error in microprocessor pipeline fall in the broad categories of timing speculation and clock skew scheduling. One approach to boost energy efficiency is by operating at a tighter frequency and detecting and correcting occasional timing errors after occurrence (e.g., Razor [17] and others [8]). Another approach aims to employ sensors for *just in-time* prediction of timing errors, and advocates time borrowing to avoid timing errors [9,16,19]. More recent works demonstrate *early prediction* of timing errors, several cycles in advance, by observing the correlation between instructions and their sensitized path delays [11, 60, 74]. In the domain of clock skew scheduling, recent works by Ye et al. and Lak et al. propose dynamic techniques to combine with timing speculation through error detection [31, 75]. However, their works do not consider real workload execution on a pipelined microprocessor, and are based on random inputs. On the other hand, the technique proposed in this dissertation–*Dynamically Adaptable Resilient Pipeline* (DARP)–overcomes this limitation through a cross-layer theme that considers real program driven circuit path sensitization. Moreover, DARP combines early prediction of timing errors with dynamic clock skew tuning to efficiently exploit the workload driven circuit delay variances.

Besides the above techniques, Rahimi et al. [56, 57] adapt the cycle-time guardband according to instruction/instruction-sequence profiling, thus improve the performance of the processor. Unlike the DARP technique, this approach aims at providing enough guardband in cycle-time, instead of taking the risk of timing-error and recovering from error after detection. Also, it does not employ timing-borrowing across pipe-stages. Moreover, the runtime adaption of this approach is based on a pre-defined lookup table. On the other hand, the online pipeline adjustment in DARP does not require any information of the workload before runtime.

Chakraborty et al. [12,13] have shown the energy-efficiency opportunity that lies in the performance-heterogeneous multicore architecture. For two topologically identical cores and a same operating frequency $f_o$: the core with the nominal frequency $f_{nom} = f_o$ is much more energy-efficient than the core with $f_{nom} \neq f_o$ and using DVFS technique to operate on $f_o$. Based on this observation, they proposed a Topologically Homogeneous and Power-performance Heterogeneous (THPH) multicore architecture to make a substantial energy-efficiency improvement compared to conventional Symmetric Multicore Processor (SMP). However, it will be intriguing to combine THPH with the dynamic pipeline adjusting techniques in DARP to even further improve the energy efficiency of the multicore processor. Moreover, THPH architecture can augment the delay variance which DARP exploits (as will be shown in Chapter 4). Therefore, DARP and THPH architecture can work in a reciprocal manner in improving the energy-efficiency of the multicore processor.

Previous works tackling PV-induced delay degradation in cache access aim at no occurrence of timing-error. For this purpose, AVICA [22] uses a uniform access latency, which is enough to cover the worst-case delay, in all cache accesses. Therefore, it suffers from unnecessary latency penalties. The variable-latency scheme proposed in previous works [47, 49] requires recording the locations of the error-sites, thus is limited on the granularity it can be applied to. Previous works [20, 51] boost the speed of slow cells in the cache, while again their hardware overhead limits the granularity they can be applied on. The error-coding techniques [28] are also prohibitive in hardware cost for severely PV-effected caches. Different from the previous works, the timing-error detection/recover technique employed in this dissertation can give a new light in tolerating the PV-induced delay degradation in cache accesses. The efficacy of this approach will be determined by the actual occurrence of the timing-error in cache access.

## 2.3 Processor Energy-Efficiency in NTC Era

*Near-Threshold Computing* (NTC) has emerged as a promising direction to improve the energy-efficiency of integrated circuits. With supply voltage $V_{dd}$ marginally higher than threshold voltage $V_{th}$, NTC exploits the super-linear $V_{dd}$-power relationship to reduce the system energy consumption. Although with promising energy efficiency, NTC also introduces various design challenges, including degraded device performance, increasing performance variations, as well as higher risk of functional failure [18, 39]. Also, leakage energy becomes dominant as supply voltage approaches the threshold value [53].

Recent works have exploited opportunities of NTC and addressed the associated challenges in various aspect. These works broadly fall into two categories: ad-hoc NTC designs and generalized design methods. In the first category, various application-specific circuits have been implemented in NTC [23, 24, 35, 44, 70, 76]. Most of these designs are targeting at ultra-low power and low performance applications, with Process-Voltage-Temperature (PVT) variation as the primary concern. The works in the second categories investigate

NTC design philosophy from different aspects. Several works [32,66,71] aim at reducing the leakage in NTC designs, which is the dominant energy component in a NTC circuit. Some other works [27,42,73] have proposed general design method to cope with severe PV-induced performance variation in NTC design. Moreover, Kutila et al. [30] have investigated the suitability of conventional 6T SRAM cell in NTC designs. None of the existing works from the above two categories have addressed the a fundamentally new design issue in a NTC processor: Migrating into the NTC region leads to a 10-100X processor frequency reduction; the memory, on the other hand, is expected to operate in the STC (Super-Threshold Computing) region, as the combined memory traffic from parallel threads remains comparable to STC. Consequently, the historic performance bottleneck lies in memory will shift into the processor core in the upcoming NTC region. This dissertation investigates this shifted performance bottleneck and proposes *Turbo Execution* to improve the energy-efficiency of a NTC microprocessor.

# CHAPTER 3

# DARP: DYNAMICALLY ADAPTABLE RESILIENT PIPELINE DESIGN IN MICROPROCESSORS

## 3.1 Background and Contributions of This Work

Rapid miniaturization of transistor devices has introduced several uncertainties in their operation. Modern microprocessor pipelines experience multiple sources of delay variations, sometimes manifesting as a timing error [7, 50, 61]. Some of the well studied sources of delay variations include process variation and aging [50, 61]. To ensure reliable operation while preserving energy efficiency under delay variation, two of the most popular techniques applied on microprocessor pipelines are timing speculation and clock skew tuning. Timing speculation reduces the guardbands to a point where errors occur and are recovered using error detection and recovery techniques like Razor [17]. Post silicon clock skew tuning is used to adjust the clock skews of the pipeline registers to allow some stages to borrow time from others [69]. Combined with timing speculation, clock skew tuning can be even more effective, as it becomes possible to configure the clock skews more aggressively to allow occasional errors, thereby improving both performance and energy efficiency [75].

Variation in delays of pipeline stages also depends on specific applications running on the microprocessor, as well as dynamic fluctuations in voltage and temperature. For instance, sensitized paths during the execution of real world applications may exhibit strikingly distinct characteristics than expected from a purely static timing analysis. However, there exist very limited works that exploit delay variance from real world applications. For example, recent works on adaptive clock skew tuning by Ye et al. and Lak et al. tackle process variation and aging based delay variation, *but do not address the delay variation from real world applications* [31, 75].

This work employs a circuit-architectural analysis to investigate and exploit the delay variation seen in sensitized circuit paths during real world application execution. This work identifies two distinct classes of these variations, driven by workloads: (a) *temporal*–delay variation within a given pipe stage during different phases of a program; and (b) *spatial*–distinct delay distributions among different pipe stages of a microprocessor. This work also exploits early error prediction, a recently proposed technique that allows to predict timing errors many cycles in advance using the instruction program counter (PC) [11,60,74]. This work combines early error prediction with clock skew tuning to propose *Dynamically Adaptable Resilient Pipeline* (DARP), which outlines a next wave of innovation in pushing the energy efficient frontier of pipelined microprocessor design.

This work makes the following contributions:

- It shows a striking temporal and spatial variance in the sensitization of critical paths in a microprocessor component, during the execution of real programs (Section 4.2). The rigorous analysis in this work integrates architectural simulation data with a gate level logic analyzer to determine critical paths and critical delays sensitized during program execution.
- It proposes DARP, a dynamically adaptable resilient pipeline. In addition to handling the well studied delay variations from process variation and aging, DARP can adapt a pipeline to the spatial and temporal delay variations from real workloads. DARP employs program phase driven early timing error prediction and dynamic frequency and clock skew adjustment to exploit workload specific delay characteristics in pipelined systems (Section 3.3).
- Using a rigorous circuit-architectural simulation (Section 4.4), combining synthesized hardware with real world application execution through architectural simulation, this work demonstrates dramatic improvement in the performance (9.4–20%) and energy efficiency (6.4–27.9%), compared to state-of-the-art timing speculation techniques (Section 3.5). DARP schemes have negligible core level power overheads of 0.84% and 3.35%, giving an energy-efficient alternative for robust pipelines.

## 3.2 Motivation

This section demonstrates the wide variance in sensitized path delays in microprocessor pipe stages during the execution of real applications. The cross-layer analysis in this work, identifies two distinct sources of this variance: *temporal* and *spatial*. Collectively, these delay variances uncover intriguing possibilities in the runtime adaptation of various pipe stages in a microprocessor, boosting performance and energy efficiency of the entire system.

### 3.2.1 Sensitized Pipe Stage Delay

There is a strong correlation between a static instruction and the paths sensitized by each of its dynamic instances [60]. The logic state of one instruction depends on the preceding instructions. During the program execution, the code path followed before the execution of a particular instruction plays a critical role in determining the specific inputs for that instruction, as well as, the preceding instruction scheduled on any pipeline stage. Since instructions are drawn from a static code, it is expected that the neighboring instructions are fairly stable (barring occasional wrong path executions where instructions enter the pipeline due to a mis-predicted branch) [60].

However, there is a wide disparity in the sensitized delays of different instructions in a microprocessor pipe stage. For example, during a particular program execution, certain instructions may experience a mere fraction of the clock cycle delay, meeting the timing with a large margin to spare. While other instructions may experience a much larger delay, with limited slack. To analyze these intriguing properties, this section employs a rigorous cross-layer methodology, outlined next.

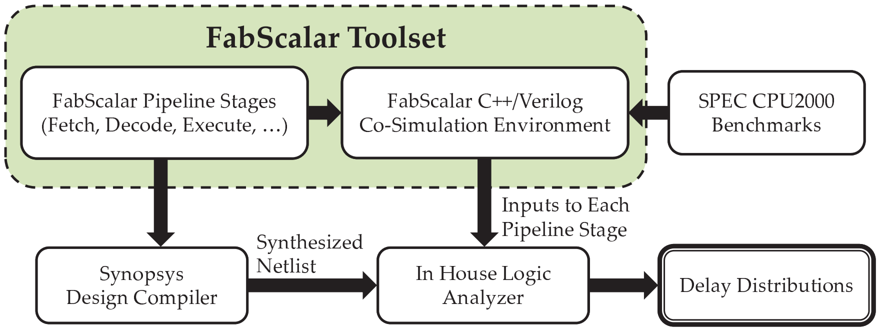
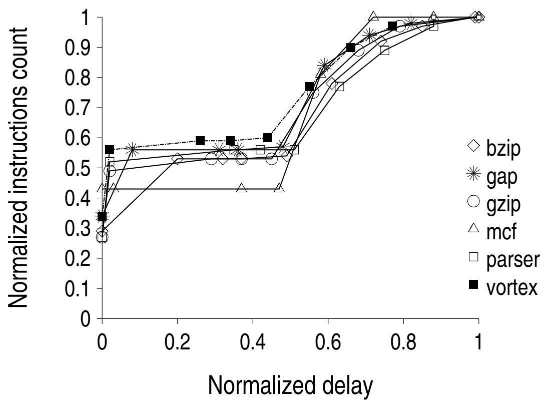
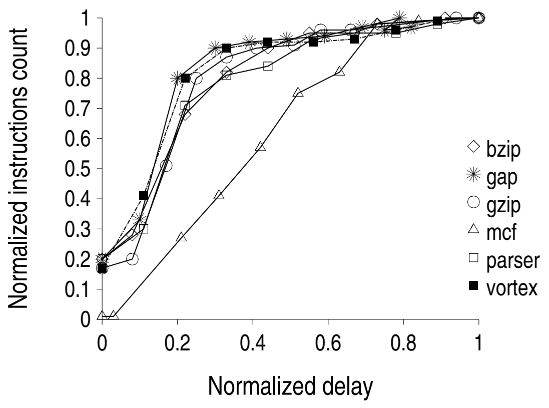

FabScalar Toolset
FabScalar Pipeline Stages (Fetch, Decode, Execute, ...)
FabScalar C++/Verilog Co-Simulation Environment
SPEC CPU2000 Benchmarks
Inputs to Each Pipeline Stage
Synthesized Netlist
Synopsys Design Compiler
In House Logic Analyzer
Delay Distributions

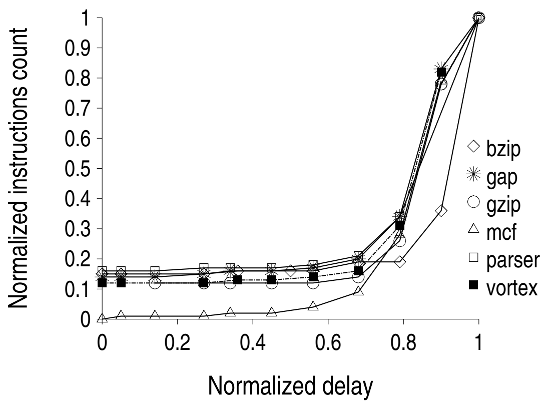
Fig. 3.1: Cross-Layer Methodology to investigate delay variance in pipe stages.



(a) Retire



(b) Execute



(c) Issue

Fig. 3.2: CDF of sensitized path delay variance in several microprocessor pipe stages.

#### 3.2.1.1 Methodology

As instructions from various workloads pass through the pipe stages, they sensitize different paths, and therefore observe different logic computation delays. To capture these characteristics, Figure 3.1 shows the rigorous cross-layer methodology of this work which combines architecture level workload simulation with circuit level timing analysis. This work uses the RTL modules, configured for the out-of-order *Core-1* configuration, from the FabScalar infrastructure [15]. The modules are synthesized using Synopsys Design Compiler to obtain a netlist of gates, Subsequently, this work performs architectural simulation of several real world applications using the FabScalar Co-Simulation environment, and extract cycle-by-cycle input vectors for various pipe stage RTL modules. This work uses these input vectors with an in-house logic analyzer to obtain the delay characteristics of various pipe stages as instructions flow through the pipeline.

#### 3.2.1.2 Results

Figure 3.2 shows the delay variance on three pipe stages. The *Retire* stage graduates instructions from the pipeline, making their changes visible outside (Figure 3.2(a)). For the *Execute* stage, this work uses the delay variance seen in the *Simple ALU* (Figure 3.2(b)). The *Issue* stage is responsible for scheduling instructions on various functional units (Figure 3.2(c)). All the three figures show the cumulative distribution function (CDF) plot for the sensitized delay from workload instructions.

These figures show a wide range of sensitized path delays in these pipe stage modules. Furthermore, various workloads also show substantially different delay profiles. On a closer inspection, these variations can be broadly characterized into two major classes:

- **Temporal:** Within a given pipe stage, different instructions in a given workload often exhibit substantial variance. For example in Figure 3.2(a), 29% instructions in *bzip* exhibit negligible delay in the *Retire* pipe stage, whereas 57% instructions in *vortex* exhibit similar characteristics.

- **Spatial:** Different pipe stages exhibit a range of delay profiles. For example, there are intriguing distinctions, as comparing *Execute* and *Issue* (Figures 3.2(b) and 3.2(c), respectively). Across different benchmarks, a vast majority of instructions exhibit 80% or higher percentage of the maximum delay in Issue, whereas only a limited fraction of instructions experience such high delay in the Execute stage. Comparing across different workloads, this spatial delay imbalance between the *Execute* and *Issue* stage is substantially more pronounced in *vortex* than in *mcf*, primarily due to more uniform delay characteristics in the *Execute* stage for *mcf*.

#### 3.2.1.3 The Source of Sensitized-delay Variation

The delay variations observed can be affected by various factors. For the execute stage, Figure 3.3 shows how the delay of the current cycle is affected by the difference between the input-data size of the immediately preceding cycle and that of the current cycle. Generally, if either cycle has a large-sized input-data, the delay of the current cycle tends to be large. On the other hand, if both cycles have small-sized input-data (e.g. Small-size to Small-size (SS) case), the current cycle tends to have a small delay. The CDF of the benchmarks in Figure 3.2(b) are shaped by their composition of the cases (Figure 3.4). The mcf benchmark in Figure 3.2(b) shows much less population in the low delay region than other benchmarks. This characteristic is consistent with Figure 3.4, as mcf has much less SS cases than other benchmarks.

Besides the impact of the input data-size on the execute stage, others stages can be affected by more complex interplay of circuit and architectural characteristics. For example, the extent of instruction dependencies can impact the sensitization of long delay in Issue, LSU and Retire stages [62]. However, in stages that generate control signals based on the sequence of instructions, isolating the factors leading to longer sensitized delay is complex.

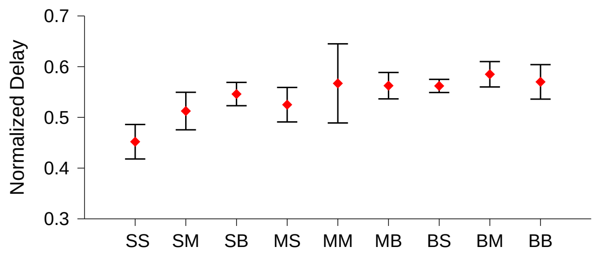


Fig. 3.3: Average delay for different input-data width transition in execute stage. Both mean and standard deviation across different benchmarks are shown. The average delay of the current cycle is measured for the transition between the input-data width of the immediately preceding cycle and that of the current cycle. SS: Small-size to Small-size; SM: Small-size to Medium-size; SB: Small-size to Big-size. Small-size: input-data width <= 8bits; Medium-size: 8bits < input-data width <= 16bits; Big-size: input-data width > 16bits.

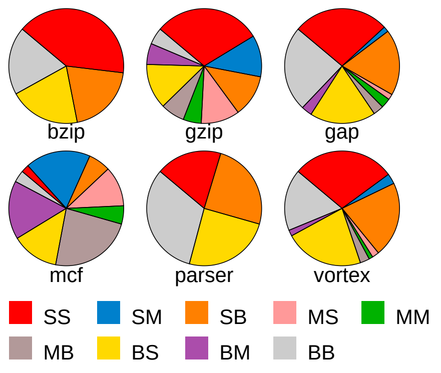


Fig. 3.4: The composition of data-width transition for the benchmarks. Using the same definition as figure 3.3, for all the transitions.

### 3.2.2 Significance

The results above indicate a wide disparity in delay sensitized from different instructions. On one hand, previous works have demonstrated *striking similarity* in delay characteristics from recurring instances of the same instruction [60, 74]. On the other hand, then is it possible to simultaneously exploit both these workload driven behaviors in an unified manner in a microprocessor pipeline? Following this genuine possibility, this work proposes the scheme named *Dynamically Adaptable Resilient Pipeline (DARP)*. DARP exploits repeatability of delay characteristics of a program by substantially tightening the operating frequency, as well as, dynamically predicting upcoming timing errors several cycles in advance. As predicted, these timing errors are subsequently avoided through a stall insertion in the pipeline, radically diminishing the penalty from error detection and correction with replay. On the other hand, spatial variances in sensitized delay in the pipe stages are seamlessly integrated by a low-overhead controller that dynamically adjusts processor frequency and clock skews in every epoch. Next section will describe the proposed schemes in details.

## 3.3 Dynamically Adaptable Resilient Pipeline

This section presents an overview of designing a DARP system and the associated design challenges.

### 3.3.1 DARP overview

Figure 3.5 shows an overview of the proposed DARP pipeline design. The figure shows two major augmentations of a microprocessor pipeline: (a) Timing Error Prediction Table (TEPT) in the decode stage coupled with error detection and recovery in every pipeline stage (Section 3.3.2); and (b) DARP controller design and its auxiliary components (CVD) in each pipeline stage to adjust the clock skew (Section 3.3.3). Collectively, the techniques in DARP work in tandem to substantially improve the power-performance characteristics of a microprocessor pipeline.

### 3.3.2 Exploiting Early Error Prediction

Four central aspects of early error prediction are described below:

**TEPT Size:** A 4K sized table dynamically manages the instruction PCs prone to cause repeated timing errors in various pipeline stages [11, 60]. In this work, this table is referred as *Timing Error Prediction Table (TEPT)*. Superficially, it may appear that effective prediction requires a large TEPT to avoid address collisions in the table. A larger TEPT can reduce false negatives. However, according to the simulations in this work, a very small set of instructions cause repeated timing violations. Hence, in this work, the TEPT size is chosen in accordance with the previous work [60], which has shown a small to moderately sized TEPT is sufficient to track instructions causing timing violations. In fact, a 4K sized TEPT leads to less than 2% false negatives from address collisions and the returns will diminish by increasing the size beyond 4K. On the other hand, a larger TEPT incurs more area and power overhead.

**Error Detection:** To detect runtime timing errors in various pipe stages, all the potentially critical paths are protected by Razor-like [17] double sampling flip-flop in their output. These critical paths are identified by a technique proposed by Lak et al. [31]. When an

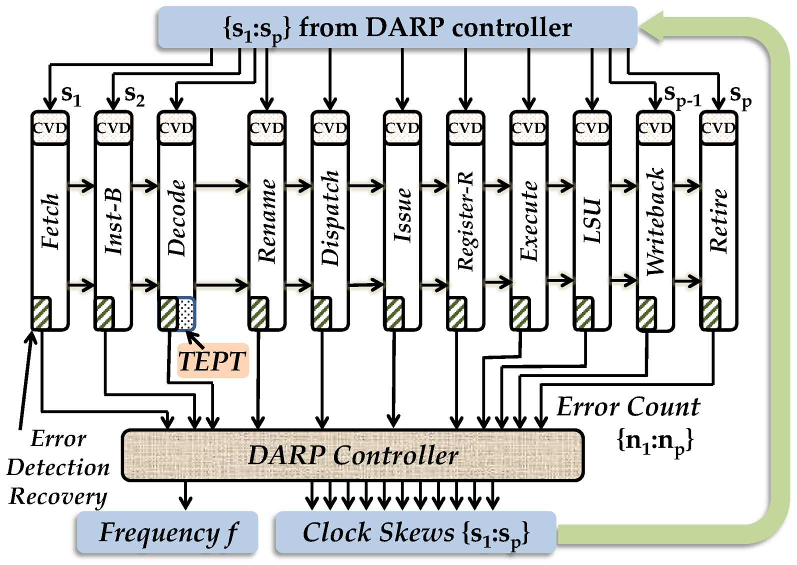


Fig. 3.5: DARP Overview.

error is detected, two actions are initiated: (a) insert the error causing instruction PC along with the relevant pipe stage information in the TEPT to avoid timing errors from the same instruction in the near future; and (b) initiate an instruction replay to recover from the fault [17]. During the repeated execution of the same instruction, timing errors are avoided as described next.

**Error Avoidance:** During the decode of an instruction [11], the TEPT will check if that instruction is likely to cause a pipeline error. If a matching entry is found, a stall signal will be propagated as the instruction proceeds in the pipeline. When the instruction enters the pipe stage where it previously caused a timing error, the stall signal is triggered, thereby allowing the instruction to occupy that pipe stage for two consecutive cycles. Forward flow of instructions is avoided for that cycle by recirculating the inputs to all other pipe stages. Consequently, timing error from that instruction is avoided, recovering a bulk of the performance loss. This is possible as pipeline stalls incur at least 10X lower penalty than an instruction replay in modern processors. During error avoidance by stalling, the timing-error flagged by Razor flip-flop is masked.

**Dynamic TEPT Management:** Entries in the TEPT are managed at runtime to exploit the workload phase behaviors. Whenever a timing error is detected, the instruction PC is inserted into the table. Recall that the detection of a timing error implies that no error was predicted from that PC before. If the table is full, as is expected after a brief table warm-up period,an eviction entry can be decided using a pseudo-LRU (least recently used) policy. Pseudo-LRU policy is widely used in modern caches to avoid the complexity of implementing a full LRU policy. Table entries are managed as a CAM (Content addressable memory), which allows the best use of the small space available.

### 3.3.3 DARP Controller Design

two components are employed to exploit the spatial imbalance among pipe stages using: (a) a system level DARP controller that dynamically determines the clock skew configurations of each pipe stage; and (b) lower level clock vernier devices (CVD) that control the clock skews of individual pipe stages. These are detailed next.

#### 3.3.3.1 DARP Controller

The role of the DARP controller is to dynamically configure the frequency of the pipeline and the clock skews in every pipe stage. This is done in hardware using the information about the timing errors in each pipe stage that are not covered by early prediction.

Algorithm 1 outlines the operation of the DARP controller. The DARP controller repeats this algorithm once in each epoch (every $N$ instructions) to dynamically adapt the pipeline frequency as well as clock skews to resolve the spatial imbalance in pipe stage delays. Each pipe stage clock skew is configured using a 3-bit skew configuration of its CVD. The input to the DARP controller is the skew configurations $\{s_1 : s_p\}$ for $p$ pipe stages, the frequency $f$, and the timing error counts $\{n_1 : n_p\}$ from the previous epoch.

**Tuning Operating Frequency**: Steps 1-7 outline the frequency tuning at each epoch. First, the pipe stages with the maximum and minimum timing errors (stages $k$ and $l$) are recognized. The operating frequency will be tighten by a factor of $(1 + \gamma)$ when $n_{min}$ is zero and $n_{max}$ is lower than a constant threshold $\rho$. A constant $\gamma$ is used To keep the implementation complexity low. Likewise, the frequency will be relaxed when $n_{min}$ exceeds a lower bound given by constant $\eta$. In summary, these steps tighten or relax the operating

**Algorithm 1** DARP

```
Input: ${s_1 : s_p}$, $f$, ${n_1 : n_p}$
Output: ${s_1 : s_p}$, $f$

1: $k \leftarrow Pipe\ stage\ with\ max\ errors$; $n_{max} \leftarrow n_k$
2: $l \leftarrow Pipe\ stage\ with\ min\ errors$; $n_{min} \leftarrow n_l$
3: if $n_{min} == 0$ && $n_{max} <= \rho$ then
4:     $f \leftarrow f * (1 + \gamma)$
5: else if $n_{min} >= \eta$ then
6:     $f \leftarrow f * (1 - \gamma)$
7: end if
8: $T \leftarrow \frac{1}{f}$
9: $n_{avg} \leftarrow avg(n_1 : n_p)$
10: $\{\triangle_1 : \triangle_p\} \leftarrow \{n_1 : n_p\} - n_{avg}$
11: for $i \leftarrow 1 : p$ do
12:     if $\triangle_i < 0$ then
13:         $s_i \leftarrow s_i - 001$
14:     else if $\triangle_i > 0$ then
15:         $s_i \leftarrow s_i + 001$
16:     end if
17: end for
18: Adjust $\{s_1 : s_p\}$ to maintain T*p cycles
```

frequency once in each epoch, based on the timing error profile of the pipeline in the previous epoch. However workload, aging and other environmental condition induced delay variations will adapt the frequency and clock skews over several epochs until they stabilize.

**Configuring Clock Skews**: Steps 9-18 illustrate the dynamic clock skew tuning. At first it calculates the set $\{\triangle_1 : \triangle_p\}$. The $i^{th}$ element in this set contains the difference between the timing errors in stage $i$ ($n_i$) and the average timing error $n_{avg}$. The pipe stages that have timing errors below average get a reduced clock skew, while the pipe stages with timing errors above average see an increase in the positive clock skew. The clock skew tuning is done using a 3-bit programmable CVD, detailed in Section 3.3.3.2. This dynamic reconfiguration of clock skews ensures that the pipe stages sensitizing higher delay paths can borrow time from those sensitizing lower delay paths. This process in turn further tightens the frequency in the next epoch. Step 18 readjusts the clock skews to ensure the total time for a $p$ stage pipeline is maintained at $T * p$ cycles, $T$ being the time period (step 8) (Time-borrowing priority goes to the most erroneous stages in case of conflicts).

#### 3.3.3.2 Clock Vernier Device (CVD)

DARP uses clock vernier devices (CVD) as the clock tuning elements (Figure 3.6 ) [38]. CVDs can generate several skew configurations based on their input bits. The latches $a$, $b$ and $c$ also store the skew configurations to be used in the next epoch. For example, a 3-bit input can can provide 8 skew configurations. If set bit 011 to represent zero skew, and perform skew increments/decrements in steps of $\delta$, then it can give the positive and negative skew configurations of $\{-3\delta, -2\delta, -\delta, 0, \delta, 2\delta, 3\delta, 4\delta\}$. These skews help dynamically lend time to pipe stages sensitizing higher delay critical paths, and borrow time from pipe stages sensitizing low delay critical paths. To manage the overhead of adding CVDs, multiple flip-flops in a pipe stage are organized in groups to share a single CVD, using the clustering algorithm proposed by Lak et al. [31].

#### 3.3.3.3 Implementation

The clock skew adjustment must be made when the pipeline is empty to avoid metastability issues [8]. Thus, at the end of every epoch, the entire pipeline is flushed before running Algorithm 1 to obtain the new frequency and skew adjustments for the new epoch. The pipeline operation is resumed only after skew adjustments are applied and the new frequency is stabilized. Several strategies are adopted to limit the overhead of this operation. First, a large enough epoch of 100K cycles is chosen to amortize the dynamic adjustment cost while

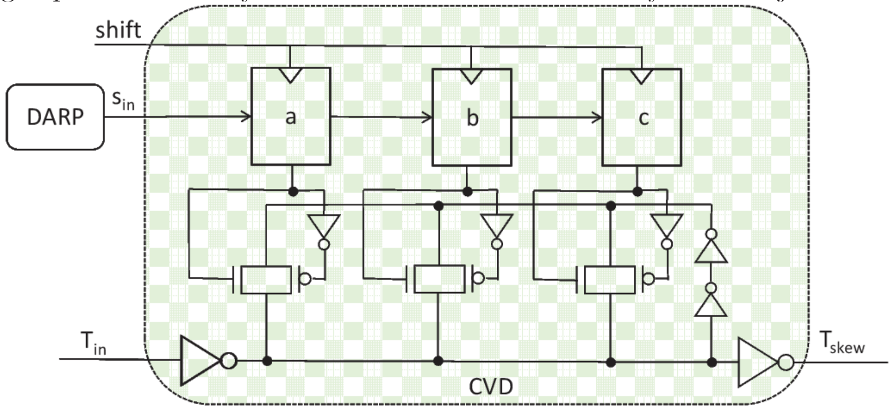


Fig. 3.6: Configuring Clock Vernier Devices using DARP.

offering enough opportunities to adapt to the workload characteristics[1]. Second, a fairly simple algorithm is used in hardware to incrementally adjust clock skews and frequency, avoiding a full blown search. Third, using a widely popular dynamic clocking technique that allows a rapid adjustment of the clock frequency [40]. Combining these strategies, the overhead of a single dynamic reconfiguration is less than 100 cycles per epoch (overhead of 0.1%).

#### 3.3.3.4 DARP for tackling delay variations due to aging, voltage and temperature fluctuations

The proposed DARP algorithm is adaptive in nature, as it is driven by real time timing errors occurring in the pipe stages. Different stages of a microprocessor pipeline may age asymmetrically over time, as thermal fluctuations are predominant in the back end of the pipeline [41]. Asymmetric aging can substantially change the critical delays of certain pipe stages leading to increased timing errors. The DARP algorithm can automatically adjust the pipeline frequency and clock skews as the processor ages, as it uses the current timing error information from each pipe stage. Similarly, if the delays of some pipe stages change due to a combination of temperature hotspots and process variation, the timing errors in that stage will increase, causing the DARP algorithm to readjust the time allotted to various stages.

## 3.4 Methodology

This section describes the cross-layer methodology in this work for power-performance trade-off analysis of DARP.

### 3.4.1 Architecture Layer

The architectural simulation in this work is based on the FabScalar infrastructure [15]. The following will provide details of the proposed core microarchitecture and specific

[1]A thread is typically scheduled for a 10ms time quantum, within which the configuration adjustments can be run 300 times assuming a 3GHz clock.

workloads used in this work.

#### 3.4.1.1 Core Microarchitecture

This work chooses FabScalar's *Core-1* configuration, with an eleven stage $p = 11$ out-of-order superscalar pipeline capable of fetching, issuing and committing 4 instructions each cycle. The core uses a two-level cache hierarchy: a 32 KB 4-way split Instruction/Data L1 cache with a latency of 1 cycle and an 8MB 16-way L2 cache with a latency of 25 cycles. The main memory access time is 240 cycles. Each pipeline stage employs a timing-error detection and *instruction replay* mechanism similar to [17] as well as the proposed DARP enhancements (Figure 3.5). This work models cycle accurate timing behavior of the proposed architecture under various schemes.

#### 3.4.1.2 Workloads

This work uses several SPEC CPU2000 integer benchmarks representing a range of real world applications, in the analysis. Each of these benchmarks is simulated for 10ms, which is a typical time quanta allotted to a thread from the operating system.

### 3.4.2 Circuit Layer

In order to evaluate the proposed approach, this work uses an in-house statistical timing analysis tool to calculate the propagation delay of each pipeline stage on every cycle (Figure 3.1). To perform gate level timing analysis, this work follows several important steps. First, it synthesizes the implementation the pipeline with the Synopsys Design Compiler. Second, to obtain propagation delay of the sensitized circuit paths, it integrates the delay characteristics from HSPICE simulation based on the Predictive Technology Model (PTM) for the 22nm node [77], with the synthesized netlist gate delays. The delay models also incorporate the effects of process variation [29]. This methodology can obtain the circuit delay characteristics of a lower technology node than using a standard cell library. Third, for SRAM based memory modules such as the L1 instruction/data cache, the branch target buffer (BTB) and the conditional branch predictor, this work uses CACTI 6.0 to get the timing information [46], and subsequently integrate it to the circuit delay.

### 3.4.3 Timing Error Simulation Methodology

This work uses a combination of two distinct strategies to simulate timing errors in the microprocessor pipeline: (1) frequency over-scaling; and (2) voltage scaling. The frequency is increased and/or the operating voltage is scaled down to a point, where timing errors begin to occur in several pipe stages. Combined together, these two strategies allow to evaluate the power-performance characteristics of the proposed schemes as well as previous works in the presence of timing errors.

## 3.5 Experimental Results

This section presents experimental results of comparing the proposed DARP system with contemporary schemes based on online clock tuning and timing speculation.

### 3.5.1 Comparative Schemes

- **Razor:** This scheme models one of the most popular techniques based on timing speculation [9,17]. Using this scheme, the operating frequency of a processor can be raised, by allowing occasional timing errors in various stages of the pipeline. These timing errors are detected using double-sampling rear end flip-flops. After error detection, an instruction replay is triggered to correct the error.
- **Online Clock Tuning with Timing Speculation (OCTTS):** This scheme models recent work to combine clock skew tuning with timing speculation [31, 75]. Instead of application driven sensitized path delays, both these schemes use random inputs to drive their clock tuning mechanisms.
- **DARP:** This scheme employs dynamic frequency tuning for each benchmark. At each operating frequency, it readjusts the clock skews of the pipe stages to best exploit the spatial imbalance (Algorithm 1).

- **DARP-Pred:** In addition to DARP, this scheme employs a 4K sized timing error prediction table (TEPT) to do early timing error prediction along the entire pipeline (Section 3.3.2). DARP-Pred can avoid most timing-errors than DARP through early prediction, thereby saving several costly instruction replays.

### 3.5.2 Performance Comparison

Figure 3.7 shows the performance of the proposed DARP schemes, compared to Razor and OCTTS. The results are normalized to the Razor scheme. By exploiting the delay variation due to topology and process variation in a general purpose out-of-order microprocessor, OCTTS can boost the application performance over Razor (19% on an average). The proposed DARP scheme further improves the performance of the system by dynamically exploiting the workload driven spatial imbalance in addition to the previous factors. DARP-Pred additionally exploits the temporal delay variance in pipe stages with early error prediction and avoidance. For example, DARP-Pred can deliver 17.9% and 20% performance improvements over OCTTS in *bzip* and *gap* benchmarks, respectively. On an average, DARP and DARP-Pred show 6.7% and 13.4% speedups over OCTTS across these benchmarks. The improvements over Razor are substantially larger.

Figure 3.8 gives further insight on the performance improvement of the DARP schemes. The first and second bar for each benchmark represents the replay cycles in the OCTTS and DARP schemes, respectively, while the third bar is for the replay and stall cycles in the DARP-Pred scheme. These schemes work at the same frequency and all these cycle numbers are normalized to the OCTTS scheme. Since DARP encounters much less timing errors, it has a much lower replay penalty than the OCTTS scheme. On the other hand, DARP-Pred converts a significant part of the replay penalty (all of it in some benchmarks, e.g. bzip) into the stall penalty, further reducing the penalty cycles from the DARP scheme. This is possible as a majority of the timing errors can be avoided using early prediction. Using only Razor at the same frequency incurs substantially higher timing errors and replay penalties, and thus is omitted from the figure. Figure 3.9 shows the false positive rate of timing-error predictor employed in DARP-Pred. The false positive rate is measured as the average occurrences of false positive in an epoch. Given such a low false positive rate shown in Figure 3.9, the performance overhead from DARP-Pred is negligible.

### 3.5.3 Energy-Efficiency and Power-Efficiency Comparison

Figure 3.10 shows the energy-efficiency of both DARP and comparative schemes, in terms of the Energy Delay Product (EDP). To search the best EDP for each scheme, this work explores discrete supply voltages in steps of 5% up to 70% of the nominal voltage. OCTTS has a notable EDP reduction, with the average value of 15.1%; while the DARP schemes show a further energy-efficiency improvement, with the average EDP reduction of 23.2% and 30%, respectively, over Razor. This energy-efficiency improvement stems from three factors: (a) workload specific skew adjustments; (b) early error prediction to avoid timing errors; and (c) delay reduction discussed in Figure 3.7. When compared against OCTTS, the DARP and DARP-Pred schemes show an average EDP reduction of 9.6% and 17.6%, respectively. Figure 3.11 shows the power-efficiency of all the schemes, in terms of Performance/Watt. Figure 3.11 shows the difference of power-efficiency across the schemes is much less than that of energy-efficiency. Performance improvement increases the power-efficiency as it reduces the leakage energy. However, timing-error recovery incurs dynamic energy-overhead. Therefore, performance improvement by tightening the pipeline will lead to limited improvement, or even deterioration (e.g. OCTTS and DARP in bzip) in power-efficiency. Finally, DARP-Pred has 17% in maximum and 5.9% in average improvement in power-efficiency over Razor.

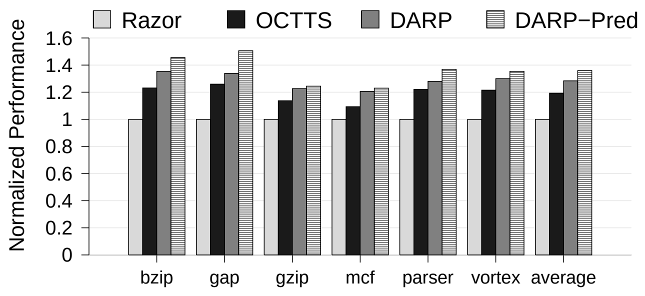


Fig. 3.7: Performance comparison normalized to Razor.

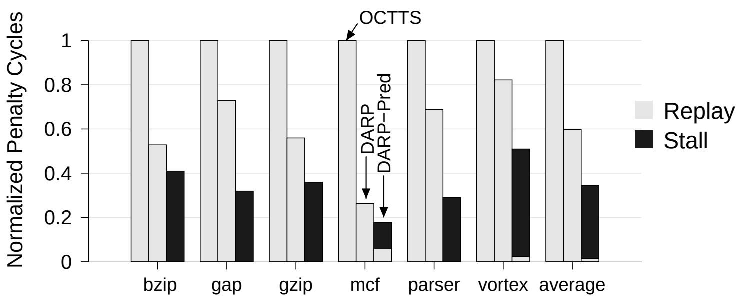


Fig. 3.8: Penalty cycles from Timing Errors.

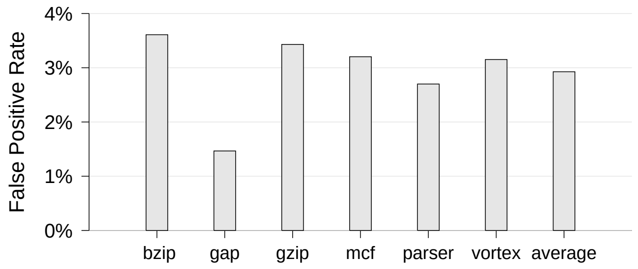


Fig. 3.9: False Positive Rate of Timing-Error Predictor in DARP-Pred.

### 3.5.4 Power Overhead of DARP

The primary power overhead in DARP comes from the counters in each stage to record timing errors, the DARP controller (Figure 3.5) and the 4K sized TEPT for DARP-Pred. These overheads are estimated relative to the OCTTS scheme. Hence the CVDs and buffers for minimum delay padding are excluded as they are already present in many modern microprocessors. The power overhead of the proposed schemes, are calculated by synthesizing the *Core-1* from the FabScalar infrastructure using the Synopsys Design Compiler with a 45nm FreePDK library. The overhead of all the hardware enhancements of the DARP pipeline are estimated relatively to the core power. Overall, the power overhead for the DARP and DARP-Pred schemes are 0.84% and 3.35%, respectively, relative to the core power. Energy efficiency and power-efficiency results presented in Figure 3.10 and Figure 3.11 include these overheads.

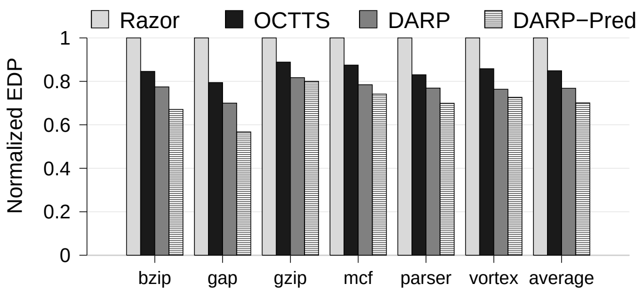


Fig. 3.10: EDP comparison normalized to Razor.

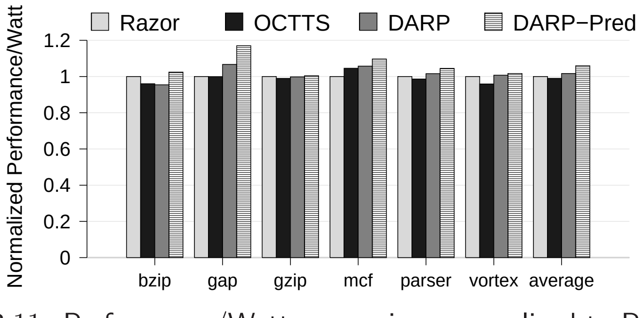


Fig. 3.11: Performance/Watt comparison normalized to Razor.

# CHAPTER 4

# DARP-MP: DYNAMICALLY ADAPTABLE RESILIENT PIPELINE DESIGN IN MULTICORE PROCESSORS

## 4.1 Background and Contributions of This Work

In the era of multicore computing, various applications running concurrently create a wide diversity in power-performance requirements. In such a computer system, fine grain control is needed to meet the steep energy efficiency demands [48]. Across the entire spectrum of microprocessor design, Dynamic Voltage and Frequency Scaling (DVFS) has been the paradigm for online energy efficiency optimization [21, 25, 33, 65]. In an increasingly diverse application pool for the multicore system, DVFS can effectively save energy consumption with minimal loss in performance.

The effectiveness of DVFS, however, is rapidly degrading as technology downscaling. One of the major sources of this inefficiency is: dynamic adaptions cannot alter the intrinsic characteristics of the circuits (e.g. gate sizes and threshold voltages). Therefore, a circuit designed for the nominal frequency will have degraded energy-efficiency if it operate on a different voltage-frequency configuration. Observing this hazard in multicore processor energy efficiency, Chakraborty et al. have proposed a Topologically Homogeneous and Power-performance Heterogeneous (THPH) multicore architecture to meet the steep energy efficiency demands in such an environment [12]. In this architecture, the cores are designed for various nominal frequencies to meet the various performance requirements in an energy-efficient manner.

Combining circuit-level dynamic adjusting techniques with the state-of-art THPH architecture can possibly build up a new frontier for the energy-efficiency of multicore processors. Previous work (DARP) [14] has revealed the striking variance in the sensitized delay within the pipeline. With dynamic frequency-scaling and clock-skewing techniques, unnecessary timing margin in pipeline stages can be removed, and the energy-efficiency of the microprocessor can be substantially improved. Moreover, as will be shown in this work, the THPH design-flow will enlarge the sensitized delay variance in the pipeline. Therefore, the DARP techniques [14] will be more effective for energy-efficiency improvement as being integrated in THPH architecture.

This work makes the following contributions:

- It shows that the power-performance heterogeneity in the state-of-art THPH architecture can augment the sensitized-delay variation (Section 4.2, Figure 4.2).
- Using an elaborate cross-layer methodology (Section 4.4), it models a THPH system equipped with DARP (Section 4.4, Figure 4.6, Table 4.1 and Table 4.2), where a larger and more up-to-date workload pool is employed for this multicore system simulation.
- It applies the DARP technique on the state-of-art THPH architecture (Section 4.3, Figure 4.3), pushing the energy-efficiency of the multicore processor to a new frontier (DARP-MP).
- It presents the advantage of DARP-MP on the performance and power/energy-efficiency in a multicore architecture (Section 4.5, Figure 4.7, 4.8, 4.9 and 4.10), as well as its limitation (Section 4.5.4). The energy-efficiency improvements of DARP-MP are 42% and 49.9%, compared against the original THPH, as well as another state-of-art multicore power-management scheme, respectively. Applying DARP on THPH architecture introduces negligible system power overhead of 4.32%.

## 4.2 Motivation

Using a cross-layer methodology, this section demonstrates the impact of circuit's nominal operation frequency upon the sensitized-delay variance within it. This flexibility in delay variance uncovers intriguing possibilities in runtime adaption of pipeline stages in a THPH multicore processor, boosting performance and energy efficiency of the entire system.

### 4.2.1 Methodology

This work employs a cross-layer methodology developed from the previous work [14], to capture the sensitized-delay within each pipe stage, as the instruction of a workload passes through the microprocessor pipeline. Figure 4.1 shows the rigorous cross-layer methodology combining architecture level workload simulation with circuit level timing analysis. It uses the RTL modules, configured for the out-of-order *Core-1* configuration, from the FabScalar infrastructure [15]. The modules are synthesized using Synopsys Design Compiler with different timing-constraints to obtain the netlists of four different nominal frequencies: $f_0, f_1, f_2, f_3$, where $f_0 = 1.2{*}f_1 = 1.5{*}f_2 = 2{*}f_3$. Subsequently, this work employs the gem5 simulator [6] to perform architectural simulation of several real world applications (SPEC CPU2006 [4]), and extract cycle-by-cycle input vectors for various pipe stage FabScalar RTL modules. Each benchmark is run for 100K cycles. Within this simulation period, it captures the characteristic of different instruction sequences as they enter separate code paths. On the other hand, hspice simulation is performed on Predictive Technology Model (PTM) [77] to obtain the propagation delay for three basic gates: inverter, 2-input nand and 2-input nor. With these gate delays, the in-house logic analyzer drives the pipe stage netlists using the input vectors extracted from gem5 simulation. Finally, for each cycle the logic analyzer obtains the delay characteristics of various pipe stages at different nominal frequencies, as instructions flow through the pipeline.

### 4.2.2 Results

Figure 4.2 shows the variation of delay distribution pattern across cores (two stages are shown here: *Execute* and *LSU*) of different nominal frequencies. Previous work [14] has shown striking spatial and temporal variances exist in pipeline stages' sensitized-delay. This work, on the other hand, reveals that these delay variances can be substantially affected by the nominal frequency of the circuit. According to Figure 4.2, there is a monotone trend in the pipe stage delay distribution pattern, as the core nominal frequency varies. Figure 4.2(a) and Figure 4.2(b) show that as nominal frequency relaxed, both *Execute* and *LSU* stages will have less paths sensitized in the critical/near-critical region. For *LSU* stage with the nominal frequency of $f_0$, 90% instructions have the sensitized-delay less than about

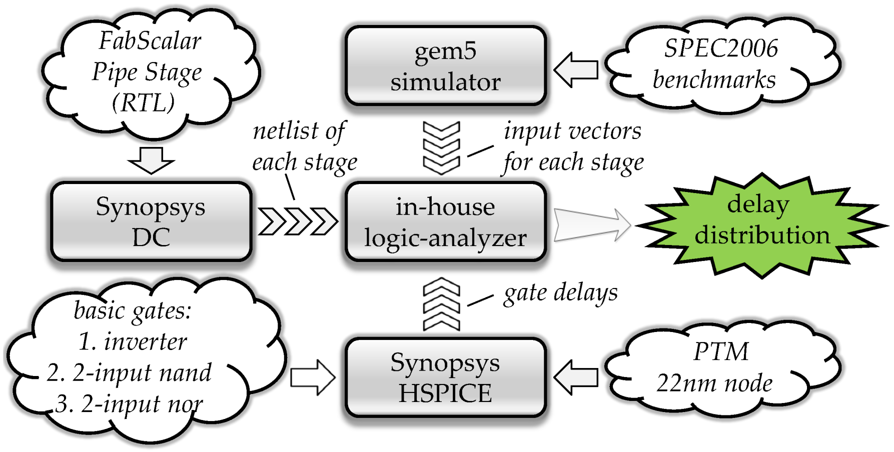
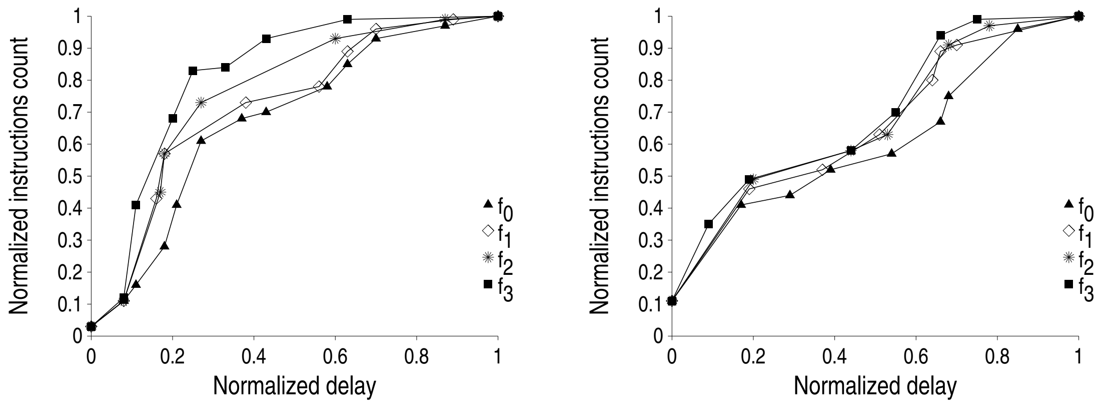


Fig. 4.1: Cross-Layer Methodology to investigate the impact of nominal frequency upon delay variance.

80% of the critical-path delay. When the nominal frequency of the LSU stage is $f_3$, 90% instructions have the sensitized-delay less than about 64% of the critical path delay. This difference comes from timing-optimization during synthesis. In conventional performance-driven design, the synthesis tool will employ various timing-optimization techniques (e.g. gate-sizing, multi-threshold) to reduce the critical path delay as far as it can. Under this optimization, the circuit will tend to have a ”critical-path wall”, where a lot of paths have the delay equal/close to the critical-path delay. On the other hand, the intrinsic characteristic of the circuit will make it less delay-balanced, if the optimization effort is relaxed. Comparing Figure 4.2(a) and Figure 4.2(b) also show that *Execute* stage is more sensitive to this optimization effort relax than *LSU* stage. In *Execute* stage, the maximum delay among 90% paths decreases from 68% to 39% of the critical delay, as the nominal frequency goes from $f_0$ down to $f_3$. This difference comes from the intrinsic topologies of these two stages: the lookup structure in the *LSU* stage intrinsically induces a lot of parallel critical-paths [36, 63].

### 4.2.3 Significance

The impact of nominal operating frequency upon the pipe stage delay distribution poses intriguing energy-efficiency opportunities for the pipeline adjusting techniques. Previous work [14] proposed DARP technique which incorporates timing-speculation and clock-



(a) Execute (b) LSU

Fig. 4.2: CDF of sensitized path delay variance across cores with different nominal frequencies.

skewing to eliminate the unnecessary timing guardband in the pipe stages. As the cores with relaxed timing-constraint during design have more spread delay distributions, applying DARP technique there will less likely to hit a "critical-path wall" than applying it on those cores with tight timing-constraint. Therefore, the cores with lower nominal frequency will have more energy-efficiency benefits than those with higher nominal frequency, as both of them employing DARP technique. The cores in the THPH architecture have various nominal frequencies. On the other hand, the cores in conventional SMP all have the same high nominal frequency. Therefore, the THPH architecture will provide extra energy-efficiency profit over the conventional SMP, assuming both of them are equipped with DARP. Section 4.3 will further discuss how DARP and THPH work together in improving the energy-efficiency of multicore processor.

## 4.3 DARP in Multicore Processor

This section briefly reviews the state-of-art THPH architecture [12,13] (Section 4.3.1), as well as the DARP technique [14] (Section 4.3.2). Then it illustrates the energy-efficiency benefit (Section 4.3.3) as well as the implementation (Section 4.3.4) to employ DARP in this architecture.

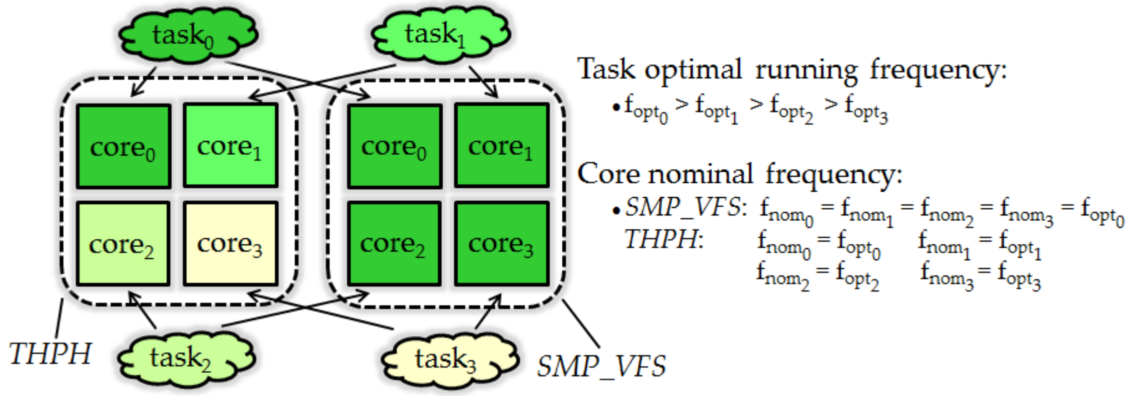
### 4.3.1 THPH Architecture

The Topologically Homogeneous Power-Performance Heterogeneous Multicore Systems (THPH) architecture uses on-chip cores that are topologically identical, but designed ground up to be power-performance optimal for separate VF (voltage-frequency) domains ($core_{0-3}$ of THPH in Figure 4.3). The THPH architecture improves the energy-efficiency of a multicore processor by exploiting the various performance needs from different tasks[1]. As different tasks have different sensitiveness to frequency downscaling, they will have different energy-optimal operating frequency ($f_{opt}$) [12, 59]. Figure 4.3 illustrates the idea of THPH, showing the $f_{opt}$ for four tasks, where $f_{opt_0} > f_{opt_1} > f_{opt_2} > f_{opt_3}$. To meet this energy-efficiency requirement, conventional SMP can be enhanced with VFS mechanism (SMP_VFS in Figure 4.3) to work on various frequencies. Then the tasks will migrate to the favored cores at runtime [59]. However, a major drawback of VFS is that it cannot change the intrinsic characteristic of the circuit (e.g, gate size and threshold voltage), thus a core designed for a higher frequency will tend to be power-hungry and energy-inefficient when working at a lower frequency [12,55]. On the other hand, different cores in the THPH architecture are specifically designed for different frequencies. Therefore, it can provide better energy-efficiency than the SMP_VFS scheme, after the tasks migrating to the favored cores [12].

### 4.3.2 DARP Technique

The DARP technique employs timing speculation, clock skewing, as well as timing error prediction to eliminate the unnecessary timing guardband in the microprocessor pipe stages [14]. Figure 4.4 illustrates the performance benefit of the DARP technique. Instead of operating at a timing-error-free frequency (with a cycle time of $T_0$), a pipeline with DARP technique applied will operate at a higher frequency (with a cycle time of $T_1$), where timing-errors will start to occur. At the occurrence of an timing error, DARP will flush the pipeline, and replay the erroneous instruction to restore the correct execution of the

[1]For a multicore system, this work uses *task* to denote the application running on each core. On the other hand, *workload* is used to denote the set of applications for the whole multicore system.



$task_0$
$task_1$
$task_2$
$task_3$
$core_0$
$core_1$
$core_2$
$core_3$
THPH
SMP_VFS
Task optimal running frequency:
• $f_{opt_0} > f_{opt_1} > f_{opt_2} > f_{opt_3}$
Core nominal frequency:
• SMP_VFS: $f_{nom_0} = f_{nom_1} = f_{nom_2} = f_{nom_3} = f_{opt_0}$
THPH: $f_{nom_0} = f_{opt_0}$ $f_{nom_1} = f_{opt_1}$
$f_{nom_2} = f_{opt_2}$ $f_{nom_3} = f_{opt_3}$

Fig. 4.3: Topologically Homogeneous Power-Performance Heterogeneous Multicore Systems (THPH) Architecture and conventional Symmetrical Multicore Processor enhanced with Voltage-Frequency Scaling (SMP-VFS). The color represents the optimal running frequency of the task and the nominal frequency of the core, as described in the figure.

workload. Consequently, how to manage the occurrence rate of the timing error becomes the primary task in DARP. For this purpose, DARP integrates two mechanisms: clock skewing and timing error prediction. Employing Clock Vernier Devices (CVD) [38], the clock skew of each pipeline stage can be adjusted. Thereby, the pipeline stages that are more likely to have timing errors ($stage_2$ in Figure 4.4) can borrow timing budget (amount of $\tau$ in Figure 4.4) from the stages that are less likely to have timing errors ($stage_1$ in Figure 4.4). Therefore, the whole pipeline will have an overall lower error occurrence. Timing-error prediction mechanism, on the other hand, extends the operation time of the pipeline stage to cover the predicted upcoming timing violation. At an occurrence of timing error, the Program Counter (PC) of the erroneous instruction, as well as the pipeline stage where the error occurs, will be recorded in a look-up table — Timing Error Prediction Table (TEPT). In the future, if another instruction instance of the same PC enter the pipeline, the associated pipeline that recorded in the TEPT will be predicted to have timing error again. As the instruction enters this stage, an extra cycle will be padded to ensure an error-free operation in that stage, while other stages in the pipeline will stall for one cycle. Collectively, a speculative pipeline stall is employed to avoid a probable pipeline flush, as enabling timing error prediction mechanism.

### 4.3.3 Potential of DARP in THPH

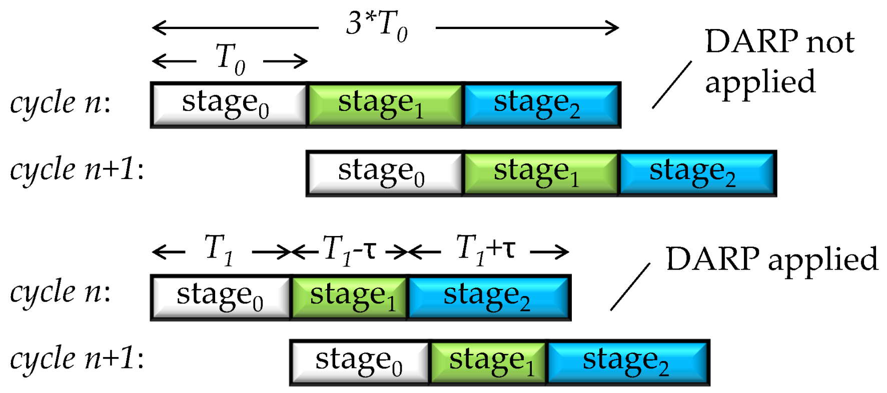


Fig. 4.4: Performance benefit of DARP Technique.

As discussed in Section 4.2, cores with different nominal operating frequencies have different degrees of freedom on pipeline adjusting. More specifically, among the topologically homogeneous cores, those with lower nominal frequency will have higher freedom for pipeline adjustment. Figure 4.5 illustrates how the benefit of pipeline adjusting technique vary across cores with different nominal operating frequencies. The *Execute* stage in FabScalar's *Core-1* configuration [15] is synthesized using four different target frequencies, where $f_0 = 1.2 * f_1 = 1.5 * f_2 = 2 * f_3$. The delay distributions for this stage are obtained from the methodology described in Section 4.2.1, with SPEC CPU2006 employed as the benchmark suite [4]. Consider the timing-speculation technique that is employed in DARP: given a same timing error rate (e.g. 5% as shown in Figure 4.5), $core_0$ (the one with frequency of $f_0$) differs significantly from $core_3$ (the one with frequency of $f_3$), in term of the frequency overscaling achieved by timing-speculation. At the error rate of 5%, $core_0$ has about 25% frequency overscaling. On the other hand, $core_3$ can overscale the frequency to more than $2X$. As incorporating clock-skewing and timing error prediction, DARP will further reduce the timing error occurrence in the pipeline. However, the core nominal frequency leads a knob to throttle the source of the potential timing errors. Therefore, the energy-efficiency improvement by applying DARP on THPH will be higher than doing so on conventional SMP architectures. This work focuses on studying the effects of dynamic adaptation (DARP technique) in a THPH style multicore processor.

### 4.3.4 Integrating DARP with THPH

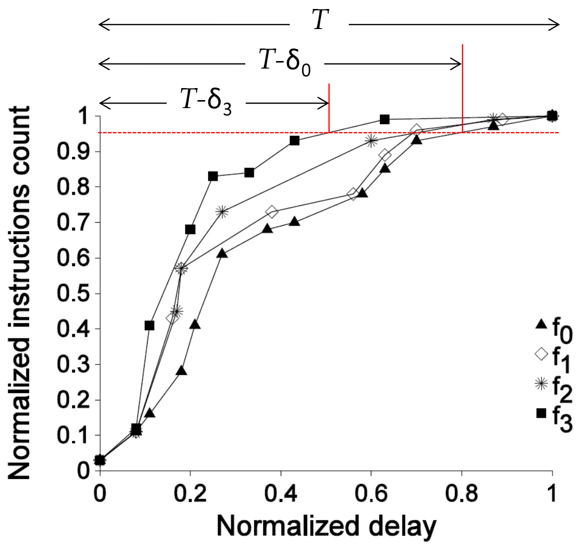


Fig. 4.5: How the benefit of DARP technique vary across cores with different nominal operating frequencies. core frequencies: $f_0 = 1.2 * f_1 = 1.5 * f_2 = 2 * f_3$.

To cooperate with task migration in THPH architecture, the TEPT, the frequency scaling factor (with respect to the nominal frequency), as well as the CVD configuration of each core, all hold copies in the shared on-chip memory (more details in Section 4.4) [12,58], to enable fast migration between cores. After a task migration, DARP controller of each core is restored from the record of previous epoch.

## 4.4 Methodology

This section describes the cross-layer methodology for power-performance trade-off analysis of DARP-MP.

### 4.4.1 Architecture Layer

The architectural simulation in this work is based on the FabScalar infrastructure [15] and gem5 simulator [6]. The following will provide details of the core/multicore architecture and specific workloads used in this work.

#### 4.4.1.1 Core Microarchitecture

This work chooses FabScalar's *Core-1* configuration, with an eleven stage $p = 11$ out-of-order superscalar pipeline capable of fetching, issuing and committing 4 instructions each cycle. The core uses a two-level cache hierarchy: a 32 KB 4-way split Instruction/Data L1 cache with a latency of 1 cycle and an 8MB 16-way L2 cache with a latency of 25 cycles. The main memory access time is 240 cycles. The pipeline equips DARP technique [14] to remove unnecessary timing guardband in each stage 4.4.

#### 4.4.1.2 Multicore Architecture

The FabScalar simulation infrastructure is limited in 1) the amount of available benchmarks; 2) modeling multicore system. Therefore, this work employs gem5 as the architectural simulation infrastructure for the multicore system. By closely examining the implementation of FabScalar pipe stages, this work identifies 1) those stages that most likely to lend/borrow time at runtime; 2) the stage-inputs that most likely to sensitize critical/near-critical delay in the stage. Then the stage-inputs are extracted cycle-by-cycle from gem5 simulation to drive the sensitized-delay analysis, which have been described in Section 4.2.1. Figure 4.6 shows the gem5-modeled 8-core system, which is similar to that presented in [12]. However, the shared L1 caches are replaced with private L1 caches to keep up with more recent design choices [67]. To support task migration among cores, a 3KB on-chip SRAM (Migration SRAM) is employed to hold the register file (RF) state results and the DARP controller records (as mentioned in Section 4.3.4), for each 4-core cluster. (For each core, 256B space is for RF [59], 512B space is for TEPT [60]. Each TEPT reserves 38bits for the clock-skew and frequency-scaling parameters.) The Migration SRAM has the access latency same as L1 cache. It has 4-banks thus can provide access to all the 4-cores in parallel. Task migration within the 4-core cluster will take 256–384 cycles to complete. The cycle number varies with the size of dirty data that needs to be saved to the Migration SRAM before the task migration.

Table 4.1 shows the system configuration, where the Voltage-Frequency for each core follows the previous work [12]. This work employs the task migration scheme described in [12], where the tasks are reassigned at each epoch (100K cycles), based on the IPC profiling of last epoch.

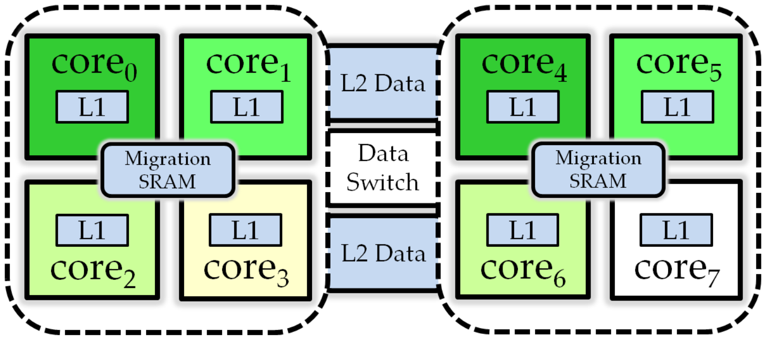


Fig. 4.6: Modeled multicore system. Different color for the core represents different core nominal frequency.

Table 4.1: Multicore configuration.

| Component | Configuration |
|---|---|
| Core 0 | 0.99V, 3.0Ghz |
| Core 1 | 0.84V, 2.4Ghz |
| Core 2 | 0.81V, 2.0Ghz |
| Core 3 | 0.81V, 1.8Ghz |
| Core 4 | 0.99V, 3.0Ghz |
| Core 5 | 0.84V, 2.4Ghz |
| Core 6 | 0.81V, 2.0Ghz |
| Core 7 | 0.81V, 1.5Ghz |
| L1 | 32KB 4-way split Instruction and Data latency= 1 cycle |
| L2 | 8MB 16-way latency=25 cycles |
| Main Memory Latency | 240 cycles |

#### 4.4.1.3 Workloads

In gem5 simulations, 16 SPEC CPU2006 (9 integer and 7 floating-point) benchmarks are randomly composed to form the workloads for the modeled 8-core system (Table 4.2). Each benchmark is simulated for 10ms, which is a typical time quanta allotted to a thread from the operating systems.

### 4.4.2 Circuit Layer

Table 4.2: Multicore Workloads.

| Workloads | Combination |
|---|---|
| W0 | libquantum, gobmk×2, hmmer×2, omnetpp, mcf, namd |
| W1 | omnetpp, sphinx3, sjeng, dealII×2, gobmk, milc, povray |
| W2 | hmmer, povray×2, milc, sjeng, gobmk, namd, mcf |
| W3 | milc, hmmer×3, h264ref, bzip2, sphinx3, sjeng |
| W4 | gcc, hmmer, sjeng, gobmk×2, libquantum, milc, povray |
| W5 | hmmer×3, lbm, libquantum, sjeng, dealII×2 |
| W6 | bzip2×3, namd, povray, libquantum×2, mcf |
| W7 | soplex, gobmk×2, dealII×2, lbm, milc, h264ref |

This work uses input vectors extracted from architectural simulation, along with a in-house statistical timing analysis tool to obtain the path sensitization delay characteristics of various pipe stages as instructions flow through the pipeline. (Figure 3.1). Several important steps are taken to perform gate level timing analysis. First, the implementation of the proposed approach is synthesized with the Synopsys Design Compiler. Second, the delay characteristics from HSPICE simulation based on the Predictive Technology Model (PTM) for the 22nm node [77], are integrated with the synthesized netlist gate delays to obtain propagation delay of the sensitized circuit paths. The delay models also incorporate the effects of process variation [29]. This methodology provide an opportunity to obtain the circuit delay characteristics of a lower technology node than using a standard cell library. Third, for SRAM based memory modules such as the L1 instruction/data cache, the branch target buffer (BTB) and the conditional branch predictor, this work uses CACTI 6.0 to get the timing information [46], and subsequently integrates it to the circuit delay.

On the other hand, this work employs Synopsys Design Compiler (DC) to evaluate the circuit power-consumption. The FabScalar core is synthesized for varies frequency-voltage constraints, along with the component usage obtained from architecture simulation results.

### 4.4.3 Timing Error Simulation Methodology

This work uses a combination of two distinct strategies to simulate timing errors in the microprocessor pipeline: (1) frequency over-scaling; and (2) voltage down-scaling. The operating frequency is increased and/or the supply voltage is scaled down until the timing errors begin to occur in several pipe stages. Combined together, these two strategies help evaluating the power-performance characteristics of the proposed schemes as well as previous works in the presence of timing errors.

## 4.5 Experimental Results

This section presents experimental results of comparing DARP-MP with original THPH as well as the combination of DARP and another state-of-art multicore power-management scheme.

### 4.5.1 Comparative Schemes For DARP-MP

- **THPH:** This scheme follows the architecture proposed in [12]: the cores are synthesized to various nominal frequencies. Task migration is employed to find the best task-assignment. No pipeline adaption is employed.
- **SVFS:** Homogeneous cores with static VFS. This scheme is loosely based on [59]. All cores are synthesized to the same nominal frequency but configured statically to work on various frequencies. Task migration is employed.
- **DARP-SMP:** Applying DARP-Pred on the conventional SMP architecture, where each core has the same nominal frequency. Task migration is employed.

### 4.5.2 Performance Comparison

Figure 4.7 shows the performance improvement when applying DARP-Pred technique on the specified multicore architectures. The results are normalized to the THPH case. SVFS and THPH have almost the same performance, as they have the topologically-identical cores and the same running frequency, as well as similar task migration mechanism. Due to the same reason discussed for Figure 3.7, both DARP-SMP and DARP-MP provide significant performance improvement over their counterparts. Moreover, due to the less balanced structures in each stage (as discussed in Section 4.2), DARP-MP has more freedom of pipeline adjustment in each core and thus gives additional performance improvement over DARP-SMP. The average performance improvements of DARP-SMP and DARP-MP over THPH are 37.1% and 54.5%, respectively.

### 4.5.3 Energy-Efficiency and Power-Efficiency Comparison

Figure 4.8 illustrates the multicore system energy-efficiency achieved by equipping the DARP-Pred technique. DARP-SMP will provide considerable energy-efficiency improvement compared to THPH, with the average EDP reduction of 22.6%. This result should be expected as DARP-Pred can substantially increase the energy-efficiency of each core. DARP-MP gives a further energy-efficiency boost compared to THPH, with the average

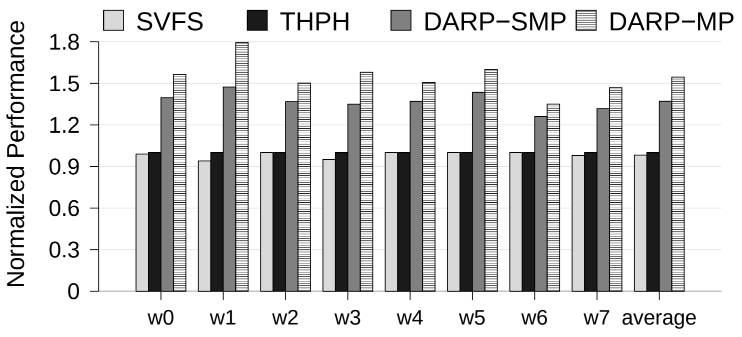


Fig. 4.7: Performance comparison normalized to THPH.

EDP reduction of 41%. Again, this extra energy-efficiency benefit comes from more freedom of pipeline adjustment in DARP-MP. Figure 4.9 illustrates this extra benefit more clearly. Applying DARP on SMP gives 31.8% energy-efficiency improvement over SVFS in average. On the other hand, the average improvement of DARP-MP over THPH is 42%.

Figure 4.10 shows the power-efficiency of all the comparative schemes. For the same reason discussed in Figure 3.11, the power-efficiency improvement provided by DARP-Pred is much limited compared to the their improvement in energy-efficiency. DARP-SMP always gives better power-efficiency than SVFS. However, due to the intrinsic power-hungry characteristics of SVFS [12], DARP-SMP gives lower power-efficiency than THPH for most workloads. DARP-MP always provides better power-efficiency than THPH, reaching an average improvement of 12.4%.

### 4.5.4 Limitations

The task migration in THPH architecture is the major limitation when applying DARP pipeline there. As shown in Section 4.3.4 and Section 4.4.1.2, an extra shared memory in each 4-core cluster needs to be employed to hold a copy of the DARP controller records. Calculated with the methodology similar to Section 3.5.4, power overhead from this memory is 3.66% and 4.32%, relative to the modeled DARP-SMP and DARP-MP architecture, respectively. This overhead has been included in the calculation in Figure 4.8 and Figure 4.10.

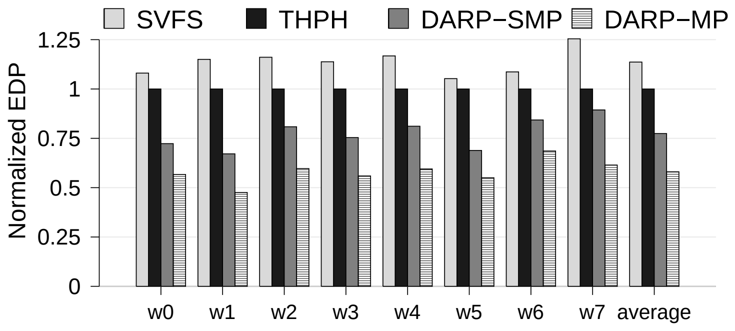


Fig. 4.8: EDP comparison normalized to THPH.

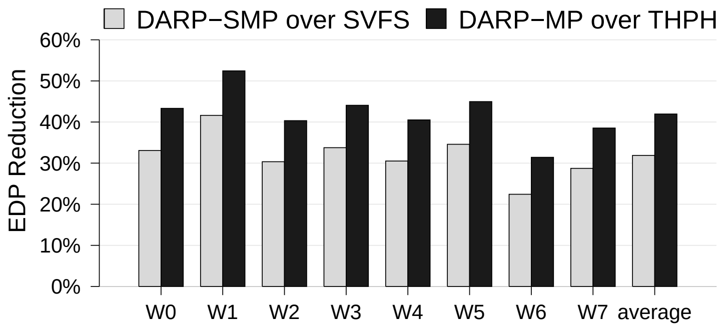


Fig. 4.9: EDP reduction when applying DARP on multicore architectures.

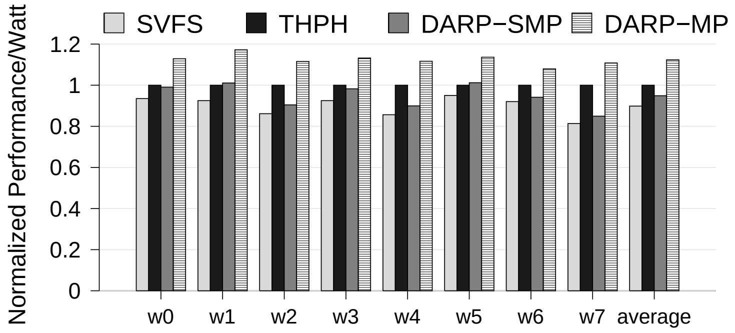


Fig. 4.10: Performance/Watt comparison normalized to THPH.

# CHAPTER 5

# OPPORTUNISTIC TURBO EXECUTION IN NTC: EXPLOITING THE PARADIGM SHIFT IN PERFORMANCE BOTTLENECKS

## 5.1 Background and Contributions of This Work

*Near-Threshold Computing* (NTC) has emerged as a promising direction to improve the energy-efficiency of integrated circuits. The NTC supply voltage $V_{dd}$ is marginally higher than its threshold voltage $V_{th}$. NTC exploits the super-linear $V_{dd}$-power relationship to reduce the system energy consumption. Inspired by this promise of better energy efficiency, many recent works have explored key challenges in NTC, such as recovering performance degradation through concurrency and tackling increased process variation [18,39,53]. While these works embody progress in NTC designs, several traditional design practices in *Super-Threshold Computing* (STC) processors are poised to be fundamentally altered in the NTC regime.

This work shows that the shift from STC to NTC can redefine the performance bottlenecks within a processor core. Migrating into the NTC region leads to a 10-100X processor frequency reduction [18, 53]. The memory, on the other hand, is expected to operate in the STC region, as the combined memory traffic from parallel threads remains comparable to STC [71]. Consequently, one of the fundamental aspects of modern computer systems—the growing gap between processor and memory speed—is now poised to reverse its direction.

This work shows that the primary performance bottlenecks in NTC processors shift from the memory to the *Long Latency Datapaths (LLD)* within the core. One class of LLDs in a modern microprocessor is the multi-cycle functional units (MFU) such as integer multiply/divide. Using a rigorous analysis, this work demonstrates that the relative performance impact from this LLD class grows by **161X**, compared to memory latency, as we transition from STC to NTC. To exploit this shift, this work proposes *opportunistic turbo execution*

(OTE)—a dynamic technique to improve the energy efficiency of NTC processors. Fundamentally, OTE dynamically speeds up LLDs by 2-5X to improve performance, while also reducing leakage energy—the major source of energy consumption in NTC systems. While conceptually intriguing, the OTE technique is not feasible in STC, as an STC pipeline already operates near its minimal delay region [39]. Consequently, OTE is fundamentally distinct from the frequency overscaling (10-20%) in some STC processors [1, 3].

Several works have focused on improving the energy efficiency of NTC circuits. One such recent technique is *Super-pipeline* that reduces the energy consumption in the NTC region [66]. By using a deeper pipeline, *Super-pipeline* transfers the circuit to the dynamic energy dominated region, thereby reducing the leakage energy. However, this circuit-level technique ignores the performance bottlenecks in an NTC processor coming from the architectural layer. Lack of this knowledge can be detrimental to the energy efficiency of the *Super-pipeline* technique. We demonstrate compelling advantages over *Super-pipeline* through a cross-layer circuit-architectural analysis, where we identify and opportunistically ameliorate the performance bottlenecks in the NTC region.

This work makes the following contributions:

- Using a cross-layer methodology combining the architecture, circuit and device layers, this work finds that the performance bottleneck shifts from the memory to the LLDs within the core, as we move from STC to NTC (Section 5.2).
- This work proposes OTE—a unique technique geared for NTC processors. OTE dynamically boosts the LLDs by 2-5X to exploit the shifting trend in performance bottlenecks in NTC processors (Section 5.3). Compared to the recently proposed Super-pipelining technique [66], OTE gives a 42.2% improvement in the NTC energy efficiency. Using synthesis followed by place and route of an ARM processor core augmented with OTE, this work observes marginal overheads in power (1.2%), wire length (0.98%), and area (0.37%), respectively (Section 5.5).

## 5.2 Motivation

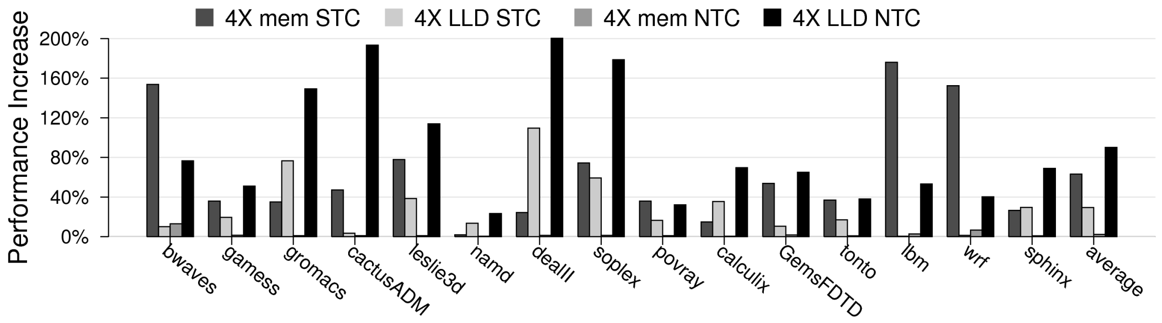


Fig. 5.1: Shifting trends in CPU performance sensitivity from the STC to the NTC regime.

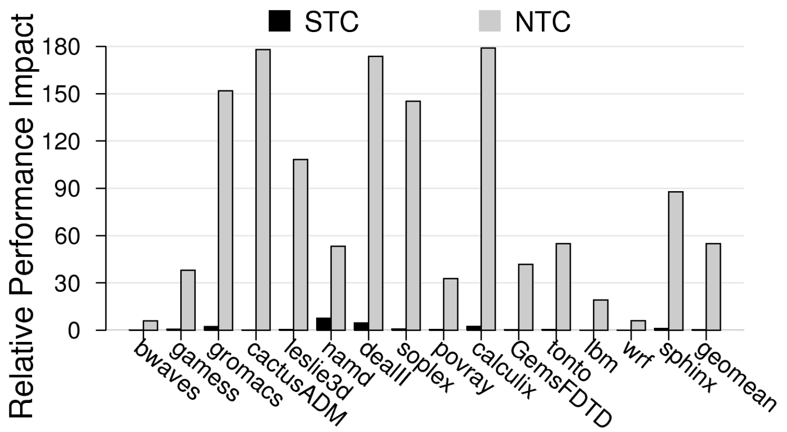


Fig. 5.2: LLD speedup over memory speedup.

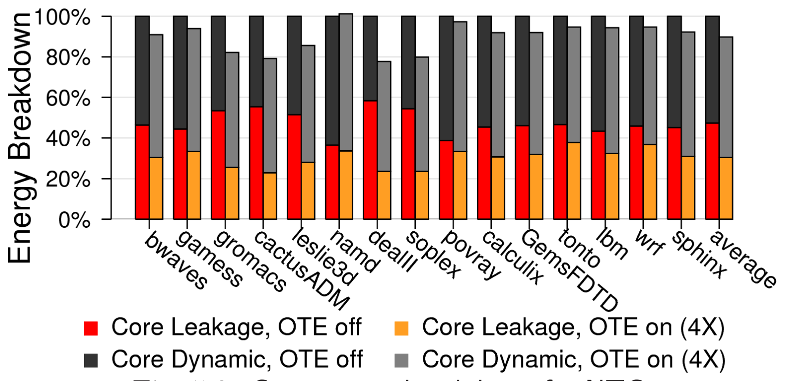


Fig. 5.3: Core energy breakdown for NTC.

This section explores shifting trends in performance bottlenecks in a processor when moving from STC to NTC. Evaluated with cycle-accurate simulation methodology (Section 5.2.1), the LLDs are poised to become the primary performance bottlenecks in NTC processors replacing the memory (Section 5.2.2). Section 5.2.3 presents the opportunity for OTE to improve the energy efficiency of NTC cores.

### 5.2.1 Methodology

Estimating the performance bottlenecks in STC and NTC presents a methodological challenge. This section briefly outlines the cross-layer approach in this work, while more details will be presented in Section 5.4. This work models a processor core similar to the ARM Cortex A15 processor [2] on the gem5 simulator [6]. SPEC CPU2006 FP benchmarks [4] are profiled to investigate the processor performance bottleneck in both STC and NTC regions. This work empirically evaluates the performance impact due to a $4X$ boost in the memory access and also in one class of LLDs–MFU. The Simpoint tool [68] is employed to pick up the representing phases of each benchmark. We assume the STC and NTC processors are clocked at 2GHz and 100 MHz [54], respectively. The memory model is given in Table 5.1, whereas the MFUs in the processor have latencies of $3 \sim 33$ cycles.

### 5.2.2 Performance Bottlenecks in NTC

#### 5.2.2.1 Comparative Speedups

Figure 5.1 shows the average sensitivity of processor performance to both memory and LLD speedup, for STC and NTC regions, respectively. The processor performance is highly sensitive to memory speedup in STC, but nearly unresponsive to it in NTC. On the other hand, performance sensitivity to LLD speedup is much higher in NTC than in STC. Figure 5.2 more clearly shows this bottleneck transition from STC to NTC. This figure presents the geometric mean of the relative performance sensitivity of LLD speedup over memory speedup. The relative performance sensitivity in STC is 0.34, on an average. In the NTC region, this sensitivity grows by **161X**, reaching 54.8 on an average.

##### 5.2.2.2 Significance

The transition observed above will potentially inaugurate a new epoch in computer system design. A vast number of works in the past few decades have been devoted to mitigate the processor-memory performance gap. In the NTC era, however, this historic bottleneck will give way to other performance barriers lying inside the processor datapath like LLDs as we have shown above. Hence it is now critical to understand and mitigate the emerging bottlenecks in upcoming NTC processors. The proposed technique *opportunistic turbo execution* (OTE), detailed in Section 5.3, aims to mitigate this bottleneck by dynamically boosting LLDs for turbo execution. A key research question is: *while OTE can deliver substantial performance boost in an NTC processor, how does it impact the processor energy consumption, as well as, the overall energy efficiency of the system?*

### 5.2.3 Energy Efficiency Perspective of OTE

Figure 5.3 presents the energy breakdown (dynamic and leakage energy) of the ARM Cortex core, with and without OTE boost. This data has been collected using an elaborate circuit-architectural methodology, which combines architectural simulation with circuit level synthesized hardware and device level HSPICE modeling, as outlined in Section 5.4. In addition to the performance gains in Figure 5.1, OTE has the potential to improve energy consumption in the NTC region by trading leakage energy for dynamic energy.

## 5.3 Opportunistic Turbo Execution

This section presents an overview of the proposed OTE technique (Section 5.3.1), the hardware support for OTE (Section 5.3.2), pipeline support for variable latency in LLDs (Section 5.3.3), and the dynamic OTE algorithm (Section 5.3.4).

### 5.3.1 OTE Overview

Figure 5.4 shows an overview of the OTE technique in a microprocessor core. The major augmentations of a microprocessor pipeline to support the normal and boosted mode for the MFUs include: voltage regulators (VR), power gates (PG), level shifters (LS), and the OTE controller that dynamically switches between the normal and the boosted mode. Collectively, these components work in harmony to boost the energy efficiency of a processor operating in the NTC region. While focusing on boosting only one class of LLDs (MFUs), the proposed OTE technique can be applied to other LLDs within the core as well (e.g., register file).

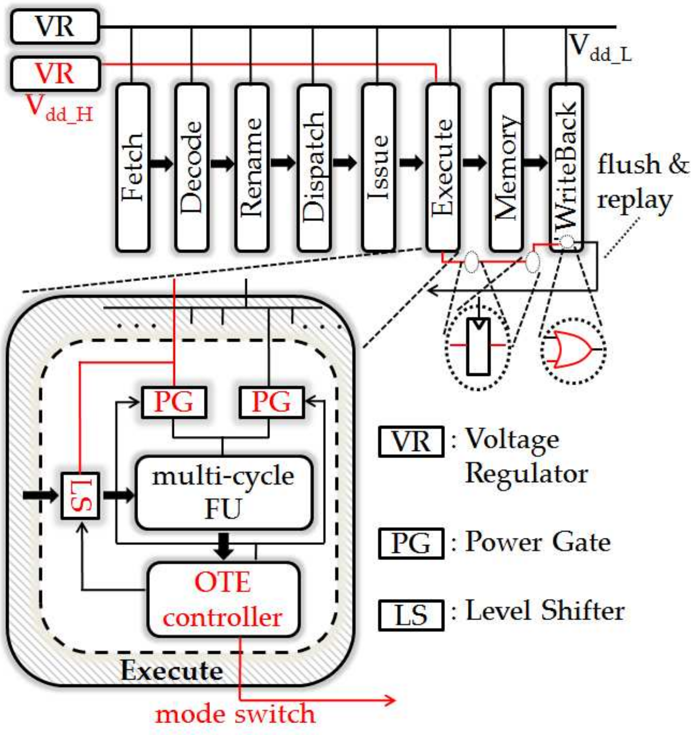


Fig. 5.4: OTE overview: the ARM Cortex A15 processor pipeline is shown, along with the major augmentations for OTE.

### 5.3.2 Supporting Two Operating Modes

OTE employs efficient voltage regulator and level shifters, as detailed next, to support two operating modes.

**Voltage Regulator**: OTE employs two power supply rails [42] with off-chip Voltage Regulators (VR) to provide dual supply voltages. $V_{dd\_H}$ in figure 5.4 is the supply voltage to support the OTE mode, in addition to the already existing voltage $V_{dd\_L}$ for the normal mode. At the 32nm technology node, the values of $V_{dd\_H}$ and $V_{dd\_L}$ are 0.6V and 0.35V, respectively, to achieve a 4X OTE boost. Depending on the decision taken by the

OTE controller (details in Section 5.3.4), the supply voltage switches between the two rails. Evaluated with a transition-time test setup similar to [42], the time to switch between the normal mode and the OTE (4X) mode can complete within one cycle (10ns) in the NTC processor.

**Level Shifter**: OTE employs the level shifter proposed in [43] to support the MFU under $V_{dd\_H}$. The level shifter is a 24 transistor circuit that allows shifting between two voltage levels, $V_{dd\_L}$ and $V_{dd\_H}$. It is controlled by select signals generated by the OTE controller (Figure 5.4). This level shifter consumes a small portion of the cycle time in OTE design [43].

### 5.3.3 Dealing with Variable Latency MFU

Enabling OTE results in variable latencies in the target LLDs (MFUs). Therefore, the pipeline micro-architecture must be augmented to allow seamless switchover between the normal and the OTE mode. There are two central aspects of this necessary modification: (a) MFU Modification and (b) Instruction Level Dependency Tracking.

**MFU Modification:** MFUs are typically pipelined in stages. Activating the OTE mode will collapse the number of pipe stages of a MFU, therefore boost its performance. To achieve this goal, a MUX circuit is added between two consecutive MFU stages. This MUX is controlled by a single-bit select indicating the OTE mode. In the OTE mode, the MUX re-directs the output of one stage directly into the input of the next stage, bypassing the intervening pipeline register. In the normal mode, this redirection is turned off, which enables only the pipeline register to drive the input of the next stage.

**Instruction Level Dependency Tracking:** Modern microprocessor pipelines are already equipped with the necessary logic to deal with variable latency operations [52]. When an instruction is scheduled on an execution unit, the issue queue logic tracks its expected completion time. For a MFU, a countdown register is used to track its completion time. Activating the OTE mode will modify all countdown registers of target MFUs to update their latencies. For example, a MFU that takes three cycles in the normal mode will complete within a single cycle under OTE (3X). At the end of this completion time, a wakeup signal is broadcasted with the corresponding tag for that instruction. Dependent instructions can then do a tag-match and grab the output, enabling correct dataflow within the pipeline.

### 5.3.4 OTE Controller

The OTE controller is responsible for dynamically switching between the normal and the OTE modes. A few key questions the OTE controller needs to decide at runtime are: (a) how much to boost the MFU (Section 5.3.4.1)? (b) when to switch from the normal to the OTE mode and vice versa (Section 5.3.4.2)? and (c) how to preserve functional correctness during the switch-over (Section 5.3.4.3)? These decisions involve a tradeoff in energy efficiency benefits and the associated overheads, outlined next.

#### 5.3.4.1 Effective Choice of OTE Boost

The ideal OTE boost can vary across different workloads. For example, in Figure 5.3, a 4X OTE boost is desirable for benchmark *dealII* due to its associated energy reduction. On the other hand, a 4X boost increases the energy for benchmark *namd* as the reduction in leakage energy is superseded by the increase in dynamic energy. However, implementing multiple performance boosts for different applications can involve a high overhead in managing multiple power rails and level shifters. Additionally, finding a minimal energy-efficiency point at runtime is impractical for real applications. Instead, this work uses a single OTE boost for the MFUs and decide the boost voltage at design time. Section 5.5.2 explores different OTE boosts to guide the choice of the OTE boost.

#### 5.3.4.2 Dynamic Control of OTE

Figure 5.1 shows that there is substantial variation in the advantage from speeding up the MFUs across different benchmarks. Even within a single benchmark, different phases can vary on their respective performance sensitivities to OTE. To effectively capture both these design aspects and improve the energy efficiency, this work explores dynamic control of OTE. The fundamental insight of dynamic control is to exploit the benchmark phase behavior at runtime. When a particular phase is exhibiting heavy utilization of the MFUs, the OTE controller switches from a normal mode to OTE. On the other hand, low utilization of the MFUs results in switchover to the normal mode. The OTE controller captures these utilization at runtime by employing the performance counters in the pipeline, which are already present in modern microprocessors. This work explores two variants of dynamic OTE, covering a range of the design space.

- *ST*: This scheme uses a single threshold for the number of cycles that the MFU is active in a given epoch[1]. If the utilization exceeds this threshold in a given epoch, the execution switches to OTE in the next epoch. Otherwise, OTE will be disabled in the next epoch.
- *HL*: This scheme has two thresholds indicating low and high watermarks, respectively. When the utilization exceeds the high watermark, the execution switches to OTE. However, unlike ST, OTE is prolonged even when the utilization is below the high watermark. When the utilization drops below the low watermark, OTE is disabled for the next epoch. This scheme essentially aims to reduce the number of transitions, and its associated overheads.

#### 5.3.4.3 Preserving Functional Correctness

A particular challenge during a mode switch arises because of alteration in the expected latencies in the MFUs (see Section 5.3.3). To maintain functionally correct execution of all instructions, the processor initiates a pipeline flush during the mode switch. Before instructions are introduced in the pipeline, the OTE controller modifies all the countdown registers for MFUs to reflect their respective latencies after the switch. Subsequently, instructions are fetched and regular processing resumes. We carefully model all circuit-architectural penalties of these aspects in our evaluation (Section 5.5).

## 5.4 Methodology

Figure 5.5 shows the extensive cross-layer methodology in this work to evaluate the proposed energy efficient NTC processor design. In this pursuit, this work combines SPICE

[1]defined period of execution

level energy characteristics of NTC and STC circuits in the device layer, synthesis and place and route based energy analysis for processor components in the circuit layer, and architectural simulation based power performance analysis in the architecture and application layers. The details for each layer are given as follows:

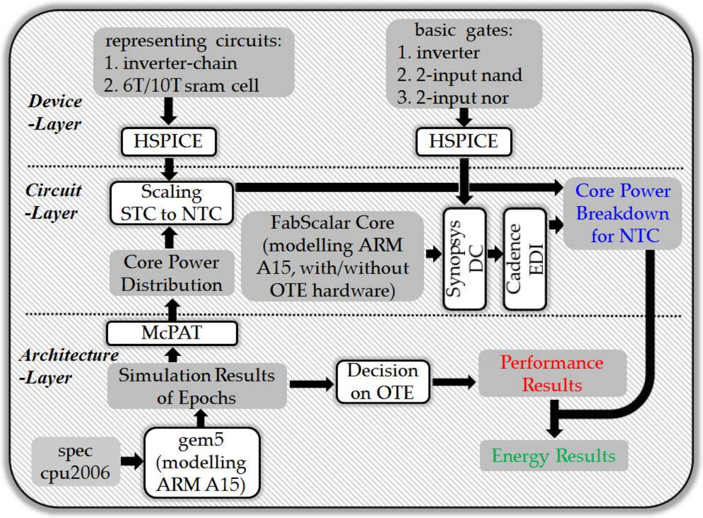

Device -Layer
representing circuits: 1. inverter-chain 2. 6T/10T sram cell
basic gates: 1. inverter 2. 2-input nand 3. 2-input nor
HSPICE
HSPICE
Circuit -Layer
Scaling STC to NTC
Core Power Distribution
FabScalar Core (modelling ARM A15, with/without OTE hardware)
Synopsys DC
Cadence EDI
Core Power Breakdown for NTC
McPAT
Architecture -Layer
Simulation Results of Epochs
Decision on OTE
Performance Results
Energy Results
spec cpu2006
gem5 (modelling ARM A15)

Fig. 5.5: Our Cross-Layer methodology.

Table 5.1: The Configuration of the ARM Cortex NTC Core.

| Parameter | Value |
|---|---|
| *ISA* | ARM |
| *Frequency* | 100MHz |
| *Re-Order Buffer* | 128 entries |
| *Issue Queue* | 64 entries |
| *Fetch/Dispatch* | 3/cycle |
| *Issue/Commit* | 8/cycle |
| *Pipeline Depth* | 15 |
| *Cacheline* | 64 Bytes |
| *L1 I-cache* | 32 KB/2-way, 1-cycle |
| *L1 D-cache* | 32 KB/2-way, 2-cycle |
| *L2 cache* | 1MB 16-way 12-cycles |
| *Memory* | B/W: 4GB/s, Latency: 22.5 $\sim$ 37.5 ns |

### 5.4.1 Architecture Layer

This work models an out-of-order processor core similar to the ARM Cortex A15 processor [2]. Table 5.1 shows the processor configuration. The MFUs in the processor have latencies of 3 $\sim$ 33 cycles. To model OTE, this work scales these latencies down by a factor of $n$, the speedup of OTE, to as low as a single cycle. The gem5 simulator [6]is employed to run 15 SPEC CPU2006 FP benchmarks [4]. The first 1 billion instructions in each benchmark are skipped to avoid the initialization period. Subsequently, each benchmark will run to its completion or 10 billion instructions, whichever happens earlier. Each benchmark is divided into epochs (epoch size = 10 million instructions) and OTE decisions are taken once per epoch. To obtain core power distribution, this work integrates the architectural simulation data with the McPAT tool [34]. McPAT uses technology parameters representing the 32nm node.

### 5.4.2 Device Layer

This work customizes the PTM 32nm technology model card for HSPICE to generate the leakage and dynamic power trends in the STC and the NTC regions [78]. The power characteristics are obtained for basic gates like nand, nor, inverter, flip-flop, and also for a 31 fanout-of-4 inverter-chain and 6T and 10T SRAM cells, as shown in Figure 5.5.

### 5.4.3 Circuit Layer

To obtain the hardware overhead of our scheme, this work adds the major augmentations for OTE (Figure 5.4) to the RTL of a FabScalar core modeled for an ARM A15 processor [15]. Then this core is synthesized using the Synopsys Design Compiler (DC) and a 45 nm reduced standard cell library using only basic gates. Place and route is performed subsequently with the Cadence Encounter tool to get a more accurate estimate of the hardware overhead including the additional power rail. The power from the device simulation of basic gates is fed to the synthesized netlist to obtain power characteristics for the 32nm technology node.

#### 5.4.3.1 STC-to-NTC Power Scaling

Scaling the entire core power from the STC to the NTC region presents a methodological challenge. HSPICE simulation of an entire processor core is computationally intense. To manage the complexity, this work adopts several steps. First, using the core power distribution from McPAT in the architecture layer, it estimates the relative power consumed by the processor components as shown in Figure 5.5. Second, it scales the STC power to NTC using the following three categories:

- *Combinational logic*: this is scaled using the STC/NTC characteristics of the canonical 31 fanout-of-4 inverter-chain as the representing circuit [53].
- *Storage elements*: this work scales the on-chip SRAM power by investigating the power scaling trend from the STC 6T SRAM cell to the NTC-friendly 10T SRAM cell [10, 72].
- *Interconnect*: McPAT does not give the power results for the interconnect within the core. However, as seen by previous works [37], this work estimates the interconnect power to be  50% of the core dynamic power. As both the interconnect power and the core dynamic power are equally affected by scaling the supply voltage, this work assumes that their relative weight remains unchanged for STC and NTC.

## 5.5 Experimental Results

This section presents a comprehensive analysis of the proposed OTE in a typical NTC processor. It briefly outlines various comparative schemes (Section 5.5.1), empirical study on OTE boost (Section 5.5.2), exploration of dynamic OTE configurations (Section 5.5.3), analysis of performance and energy efficiency (Section 5.5.4), and the overhead and limitation of the proposed schemes (Section 5.5.5).

### 5.5.1 Comparative Schemes

- **Super-pipeline:** This scheme uses a deeper pipeline to transfer the circuit to the dynamic energy dominated region, thereby reducing the leakage energy [66]. Seok et al. showed a 50% depth increase in a single pipe stage. To capture the maximum benefit from Super-pipeline, we optimistically model a 50% increase in pipeline depth in the entire processing core.
- **Fixed-OTE:** This is the static—always on—OTE scheme.
- **Dynamic-OTE:** In this scheme, OTE is dynamically controlled as described in Section 5.3.4.2.

### 5.5.2 Choice of OTE Boost

Figure 5.6 presents the design space exploration with four possible OTE boosts for the Fixed-OTE scheme. The figure shows a trend of diminishing improvements in energy efficiency as increasing the boost strength to *4X*. Eventually, at *5X* OTE boost, there is a degradation in energy efficiency, as the power consumption to support a 5X boost masks the performance gain through it. This work chooses a *4X* boost at design time.

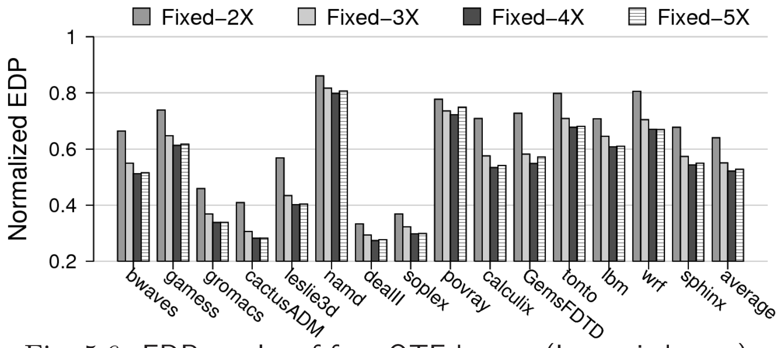


Fig. 5.6: EDP results of four OTE boosts (Lower is better).

### 5.5.3 Dynamic OTE Configuration

Figure 5.7 presents the EDP results for three variants of the Dynamic-OTE scheme (see Section 5.3.4.2). In all schemes, the runtime utilization of LLDs is monitored for an epoch length of 10M instructions. For the ST mode, OTE is activated when this utilization is above 5M cycles. For HL(10%) and HL(20%), the high and low watermarks are set at at 10% and 20%, respectively, above and below 5M cycles. Figure 5.7 shows that ST and HL(10%) perform comparably, but a larger spread in watermarks degrades the energy efficiency of HL(20%). Henceforth, this work uses the ST configuration of Dynamic OTE.

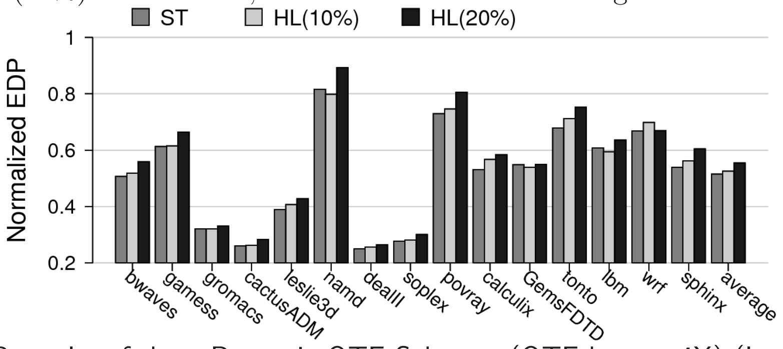


Fig. 5.7: EDP results of three Dynamic-OTE Schemes (OTE boost=4X) (Lower is better).

### 5.5.4 Performance and Energy Efficiency

Figures 5.8 and 5.9 show the performance improvement and energy reduction of all the three schemes outlined in Section 5.5.1. These are calculated with respect to a *Baseline* NTC core having no OTE or Super-pipelining. The proposed schemes are substantially more effective to drive performance gains, as they can exploit the emerging architectural bottlenecks in the NTC processor. For example, the performance of Dynamic OTE is 96.1%-178% higher than super-pipeline, across all the benchmarks. This comparatively poor result in super-pipeline stems from the fact that the gain in clock frequency afforded through it cannot efficiently exploit the bottleneck from LLDs. The proposed schemes also offer substantially higher energy reduction, by reducing the leakage energy more efficiently by optimizing the delay in LLDs. The average energy reduction of Fixed-OTE and Dynamic-OTE are 20.7X and 31.3X, respectively, compared to super-pipeline.

Collectively, gains in both performance and energy consumption leads to a significant improvement in energy efficiency of the system. Figure 5.10 shows the comparison of energy efficiency using the energy-delay product as the metric. The EDP of Fixed-OTE and Dynamic-OTE are 41.7% and 42.2% lower than the super-pipeline scheme. However, super-pipeline also outperforms our OTE schemes in some benchmarks like *namd*. The *namd* benchmark performs worse, as it is not sensitive to increase in pipeline latency and has little usage of the MFUs.

### 5.5.5 Overhead and Limitation

The on-chip overhead of the proposed schemes comes from two sources: control for the

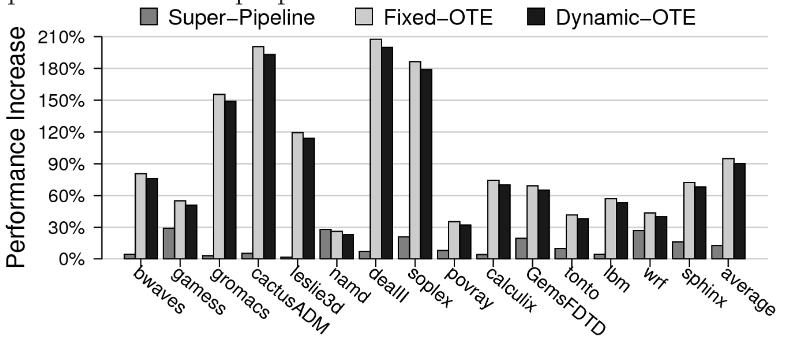


Fig. 5.8: Performance improvement (Higher is better).

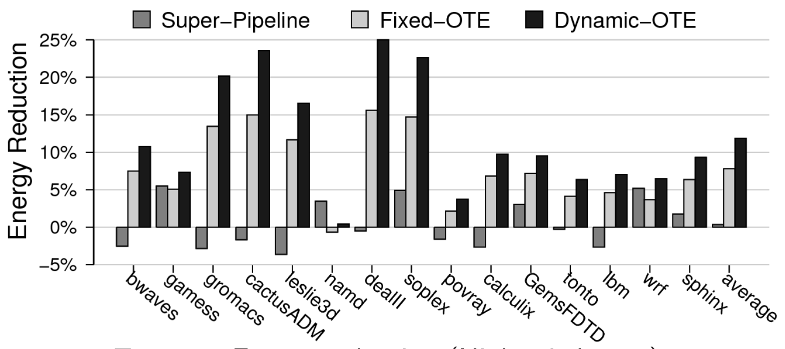


Fig. 5.9: Energy reduction (Higher is better).

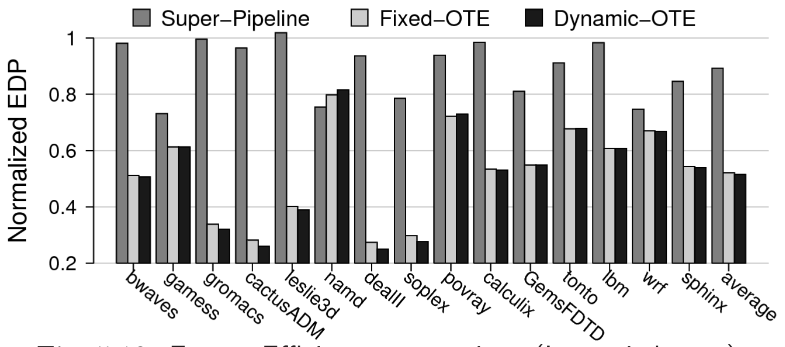


Fig. 5.10: Energy-Efficiency comparison (Lower is better).

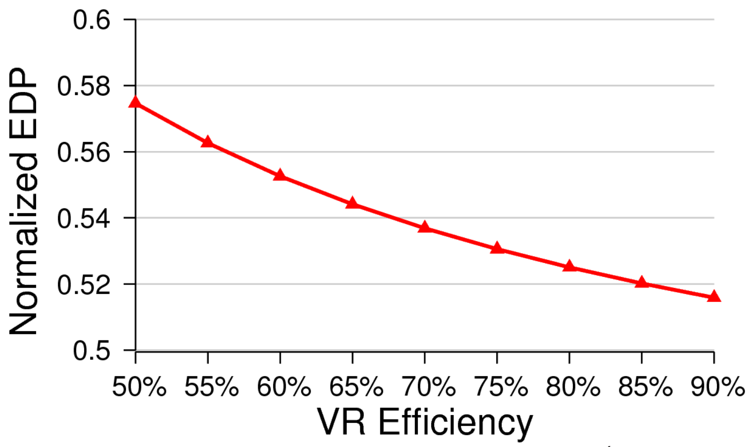


Fig. 5.11: Impact of Voltage-Regulator efficiency on the EDP (normalized to the baseline) achieved by Dynamic-OTE.

LLDs and control for the pipeline (Section 5.3). Using the circuit methodology detailed in Section 5.4.3, synthesis followed by place and route are performed on the ARM core augmented with the proposed OTE scheme. OTE only leads to marginal overheads in power (1.2%), wire length (0.98%), and area (0.37%), respectively.

**Efficacy**: The efficacy of the OTE scheme relies on the VR efficiency. Figure 5.11 illustrates that the normalized EDP of the Dynamic-OTE scheme deteriorates from 51.6% to 57.5%, as the VR efficiency decreases from 90% to 50%.

# CHAPTER 6

# CONCLUSION

This dissertation has addressed various critical challenges in microprocessor design, associated with rapid miniaturization in semiconductor devices. This work proposes novel techniques to efficiently tolerate the timing-errors in a microprocessor. Also, it explores innovative design paradigms in NTC era to promote energy efficiency in processors.

This work presents a novel runtime approach (DARP) to exploit the variations in sensitized path delays among various pipe stages in a modern microprocessor design. As running real world workloads, the microprocessor pipeline manifests striking variations in sensitized path delay. The proposed DARP technique exploits these delay variations to improve the energy efficiency of the microprocessor. Two pillars of the proposed core microarchitecture are early prediction of timing errors from program phases, and exploiting a low-overhead controller for frequency tuning and clock skew adjustments. Through a rigorous circuit-architectural infrastructure, the proposed technique demonstrates significant improvements in the performance (9.4–20%) and energy efficiency (10–28.6%), compared to state-of-the-art techniques.

This work also combines DARP with a state-of-art THPH architecture to build a new frontier for the energy-efficiency of the multicore system. Applying DARP on each core of THPH architecture will improve its energy efficiency, individually. On the other hand, THPH architecture can augment the delay variance which DARP exploits. The average EDP reductions of this combination scheme are 42% and 49.9%, compared to original THPH and another state-of-art multicore power-management scheme, respectively.

As transitioning from STC to NTC, this work identifies a shifting trend in performance bottlenecks in a microprocessor pipeline. One of the fundamental aspects in a STC computer system — the growing gap between processor and memory speed, will be largely overshadowed by the new performance bottleneck in a NTC system: the Long-Latency

Datapaths within the core. Observing this intriguing change, this work proposes OTE which dynamically boosts Long-Latency Datapaths in the processor pipeline. Evaluated by rigorous circuit-architectural analysis, the proposed approach demonstrates an average of 42.2% improvement in energy efficiency over a recently proposed technique, across a range of benchmarks.

# REFERENCE

[1] *AMD Turbo Core Technology.* http://www.amd.com/en-us/innovations/software-technologies/turbo-core.

[2] *ARM Cortex-A Series.* http://www.arm.com/products/processors/cortex-a/.

[3] *Intel Core-i7 Processors.* http://www.intel.com/content/www/us/en/processors/core/core-i7-processor.html.

[4] *SPEC CPU2006 benchmarks.* http://www.spec.org/cpu2006/.

[5] A.Agarwal, B.C.Paul, S.Mukhopadhyay, and K.Roy. Process Variation in Embedded Memories: Failure Analysis and Variation Aware Architecture. In *IEEE J. Solid-State Circuits* (2005), vol. 40, pp. 1804–1814.

[6] Binkert, N. and others The gem5 simulator. *SIGARCH Computer Architecture News 39*, 2 (Aug. 2011), 1–7.

[7] Borkar, S. Design Perspectives on 22nm CMOS and Beyond. In *Proc. of 46th Proc. of DAC* (2009), pp. 93–94.

[8] Bowman, K. and others Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. *J. of Solid-State Circ. 44*, 1 (2009), 49–63.

[9] Bowman, K. and others Circuit techniques for dynamic variation tolerance. In *Proc. of DAC* (2009), pp. 4–7.

[10] Calhoun, B., and Chandrakasan, A. A 256-kb 65-nm Sub-Threshold SRAM Design for Ultra-Low-Voltage Operation. In *JSSC* (March 2007), vol. 42, pp. 680–688.

[11] Chakraborty, K. and others Efficiently Tolerating Timing Violations in Pipelined Microprocessors. In *Proc. of DAC* (2013), no. 102.

[12] Chakraborty, K., and Roy, S. Topologically Homogeneous Power-Performance Heterogeneous Multicore Systems. In *Proc. of DATE* (Mar. 2011), pp. 1–6.

[13] Chakraborty, K., and Roy, S. Architecturally Homogeneous Power-Performance Heterogeneous Multicore Systems (in press). *TVLSI 21*, 4 (April 2013), 670–679.

[14] Chen, H. and others DARP: Dynamically Adaptable Resilient Pipeline Design in Microprocessors. In *Proc. of DATE* (2014), pp. 1–6.

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

[16] Choudhury, M. R. and others TIMBER: Time borrowing and error relaying for online timing error resilience. In *Proc. of DATE* (2010), pp. 1554–1559.

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

[18] Dreslinski, R. G. and others Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. *Proceedings of the IEEE 98*, 2 (2010), 253–266.

[19] Ghasemazar, M., and Pedram, M. Minimizing the Energy Cost of Throughput in a Linear Pipeline by Opportunistic Time Borrowing. In *Proc. of ICCAD* (2008).

[20] Gregg, J., and Chen, T. W. Post Silicon Power/Performance Optimization in the Presence of Process Variations Using Individual Well-Adaptive Body Biasing. In *TVLSI* (Mar 2007), vol. 15, pp. 366–376.

[21] Herbert, S., and Marculescu, D. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *ISLPED* (2007), pp. 38–43.

[22] Hong, S., and Kim., S. AVICA: An Access-time Variation Insensitive L1 Cache Architecture. In *DATE* (2013).

[23] Hsieh, W., and Hwang, W. All Digital Linear Voltage Regulator for Super- to Near-Threshold Operation. *TVLSI 20*, 6 (2012), 989–1001.

[24] Hsu, S. and others A 280mV-to-1.1V 256b Reconfigurable SIMD Vector Permutation Engine with 2-Dimensional Shuffle in 22nm CMOS. In *ISSCC* (2012), pp. 178–180.

[25] Isci, C. and others An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In *Proc. of MICRO* (2006), pp. 347–358.

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

[27] Karpuzcu, U. R. and others EnergySmart: Toward energy-efficient manycores for Near-Threshold Computing. In *HPCA* (2013), pp. 542–553.

[28] Kim, J. and others Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding. In *Proc. of MICRO* (2007).

[29] Kothawade, S. and others Analysis of Intermittent Timing Fault Vulnerability. *Microelectronics Reliability 52*, 7 (July 2012), 1515–1522.

[30] Kutila, M. and others Simulations on 130 nm technology 6T SRAM cell for Near-Threshold operation. In *Proc. of ISCAS* (2014), pp. 1211–1214.

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

[32] Lee, S. K. and others Evaluation of voltage stacking for near-threshold multicore computing. In *ISLPED* (2012), pp. 373–378.

[33] Li, H. and others Combined circuit and architectural level variable supply-voltage scaling for low power. *TVLSI 13*, 5 (2005), 564–576.

[34] Li, S. and others McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proc. of MICRO* (2009), pp. 469 –480.

[35] Liu, T., and Rabaey, J. M. A 0.25 V 460 nW Asynchronous Neural Signal Processor With Inherent Leakage Suppression. *J. of Solid-State Circ. 48*, 4 (2013), 897–906.

[36] Lu, S.-L. Speeding Up Processing with Approximation Circuits. *ICS 37*, 3 (2004), 67–73.

[37] Magen, N. and others Interconnect-Power Dissipation in a Microprocessor. In *Proc. of SLIP* (2004), pp. 7–13.

[38] Mahoney, P. and others Clock distribution on a dual-core, multi-threaded Itanium®-family processor. In *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International* (Feb 2005), pp. 292–599 Vol. 1.

[39] Markovic, D. and others Ultralow-Power Design in Near-Threshold Region. *Proceedings of the IEEE 98*, 2 (2010), 237–252.

[40] McNairy, C., and Bhatia, R. Montecito: A Dual-Core, Dual-Thread Itanium Processor. *IEEE Micro 25*, 2 (2005), 10–20.

[41] Mesa-Martinez, F. J. and others Power model validation through thermal measurements. In *Proc. of ISCA* (2007), pp. 302–311.

[42] Miller, T. N. and others Booster: Reactive Core Acceleration for Mitigating the Effects of Process Variation and Application Imbalance in Low-Voltage Chips. In *HPCA* (2012), pp. 1–12.

[43] Mohanty, S. P., and Pradhan, D. K. ULS: A dual-$V_{th}$/high-kappa nano-CMOS universal level shifter for system-level power management. *JETC 6*, 2 (2010).

[44] Moon, J. and others A 0.4-V, 90 $\sim$ 350-MHz PLL With an Active Loop-Filter Charge Pump. *TCAS 61-II*, 5 (2014), 319–323.

[45] Mukhopadhyay, S. and others Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. *TCAD 24*, 12 (dec. 2005), 1859 – 1880.

[46] Muralimanohar, N. and others Architecting Efficient Interconnects for Large Caches with CACTI 6.0. *IEEE Micro 28*, 1 (2008), 69–79.

[47] Mutyam, M. and others Process Variation-Aware Adaptive Cache Architecture and Management. In *IEEE Trans. Computers* (Jul 2009), vol. 58.

[48] Ogras, Ü. Y. and others Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip. In *Proc. of DAC* (2007), pp. 110–115.

[49] Ozdemir, S. and others Yield-Aware Cache Architectures. In *Proc. of MICRO* (2006), pp. 15–25.

[50] Pan, S. and others IVF: Characterizing the vulnerability of microprocessor structures to intermittent faults. In *Proc. of DATE* (2010), pp. 238–243.

[51] Pan, Y. and others Selective Wordline Voltage Boosting for Caches to Manage Yield under Process Variations. In *DAC* (2009).

[52] Patterson, D. A., and Hennessy, J. L. *Computer Organization and Design*, 4 ed. Morgan Kaufmann, 2009.

[53] Pinckney, N. R. and others Assessing the performance limits of parallelized near-threshold computing. In *DAC* (2012), pp. 1147–1152.

[54] Pu, Y. and others Misleading energy and performance claims in sub/near threshold digital systems. In *Proc. of ICCAD* (2010), pp. 625–631.

[55] Rabaey, J. M. and others *Digital Integrated Circuits*, 2 ed. Prentice Hall, 2003.

[56] Rahimi, A. and others Analysis of instruction-level vulnerability to dynamic voltage and temperature variations. In *2012 Design, Automation & Test in Europe Conference*

*& Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012* (2012), pp. 1102–1105.

[57] Rahimi, A. and others Application-Adaptive Guardbanding to Mitigate Static and Dynamic Variability. *IEEE Trans. Computers 63*, 9 (2014), 2160–2173.

[58] Rangan, K. K. and others Thread motion: fine-grained power management for multi-core systems. In *36th International Symposium on Computer Architecture (ISCA 2009), June 20-24, 2009, Austin, TX, USA* (2009), pp. 302–313.

[59] Rangan, K. K. and others Thread motion: Fine-grained Power Management for Multi-core Systems. In *Proc. of ISCA* (2009), pp. 302–313.

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

[61] Sarangi, S. and others VARIUS:A Model of Process Variation and Resulting Timing Errors for Microarchitects. *IEEE Trans. on Semiconductor Manufacturing 21* (2008), 3 –13.

[62] Sartori, J., and Kumar, R. Compiling for energy efficiency on timing speculative processors. In *Proc. of DAC* (2012), pp. 1301–1308.

[63] Sartori, J., and Kumar, R. Compiling for energy efficiency on timing speculative processors. In *Proc. of DAC* (2012), pp. 1301–1308.

[64] S.Borkar. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. In *IEEE, MICRO* (Nov 2005), vol. 25, pp. 10–16.

[65] Semeraro, G. and others Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In *Proc. of MICRO* (2002), pp. 356–367.

[66] Seok, M. and others Pipeline strategy for improving optimal energy efficiency in ultra-low voltage design. In *DAC* (2011), pp. 990–995.

[67] Shah, M. and others Sparc T4: A Dynamically Threaded Server-on-a-Chip. *IEEE Micro 32*, 2 (2012), 8–19.

[68] Sherwood, T. and others Automatically characterizing large scale program behavior. In *ASPLOS* (2002), pp. 45–57.

[69] Tadesse, D. and others AutoRex: An automated post-silicon clock tuning tool. In *Proc. of ITC* (2009), pp. 1–10.

[70] Tokunaga, C. and others A graphics execution core in 22nm CMOS featuring adaptive clocking, selective boosting and state-retentive sleep. In *ISSCC* (2014), pp. 108–109.

[71] Wang, H. and others Improving platform energy: chip area trade-off in near-threshold computing environment. In *Proc. of ICCAD* (2013), pp. 318–325.

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

[73] Xie, Q. and others Variability-aware design of energy-delay optimal linear pipelines operating in the near-threshold regime and above. In *Proc. of GLSVLSI* (2013), pp. 61–66.

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

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

[76] Zhang, F. and others Design of a 300-mV 2.4-GHz Receiver Using Transformer-Coupled Techniques. *J. of Solid-State Circ. 48*, 12 (2013), 3190–3205.

[77] Zhao, W., and Cao, Y. New Generation of Predictive Technology Model for sub-45nm Early Design Exploration. *IEEE Transactions on Electron Devices 53*, 11 (2006), 2816 –2823.

[78] Zhao, W., and Cao, Y. Predictive Technology Model, June 2012.

# VITA

## Hu Chen

### Published Journal Articles

- DARP-MP: Dynamically Adaptable Resilient Pipeline Design in Multicore Processors. Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2015 ACM Transactions on Design Automation of Electronic Systems (TODAES).*

### Published Conference Papers

- Opportunistic turbo execution in NTC: exploiting the paradigm shift in performance bottlenecks. Hu Chen, Dieudonne Manzi, Sanghamitra Roy, Koushik Chakraborty. *2015 IEEE/ACM Design Automation Conference (DAC).*
- DARP: Dynamically Adaptable Resilient Pipeline Design in Microprocessors. Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2014 IEEE/ACM Design, Automation and Test in Europe (DATE).*
- Exploiting Static and Dynamic Locality of Timing Errors in Robust L1 Cache Design. Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2014 IEEE International Symposium on Quality Electronic Design (ISQED).*
- SwiftGPU: Fostering Energy Efficiency in a Near-Threshold GPU Through Tactical Performance Boost. Prabal Basu, Hu Chen, Shamik Saha, Koushik Chakraborty, Sanghamitra Roy. *2016 IEEE/ACM Design Automation Conference (DAC).*
- Synergistic Timing Speculation for Multi-threaded Programs. Atif Yasin, Jeff Zhang, Hu Chen, Sanghamitra Roy, Koushik Chakraborty. *2016 IEEE/ACM Design Automation Conference (DAC).*