UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RAFAEL BILLIG TONETTO

A Reliability- and Variation-Aware Methodology for Improved Processor Designs for the Edge Computing Domain

Thesis presented in partial fulfillment of the requirements for the degree of Doctor of Computer Science

Advisor: Prof. Dr. Gabriel Nazar
Co-advisor: Prof. Dr. Antonio Carlos Schneider Beck Filho

Porto Alegre
September 2023

CIP — CATALOGING-IN-PUBLICATION

Billig Tonetto, Rafael

A Reliability- and Variation-Aware Methodology for Improved Processor Designs for the Edge Computing Domain / Rafael Billig Tonetto. – Porto Alegre: PPGC da UFRGS, 2023.

120 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2023. Advisor: Gabriel Nazar; Co-advisor: Antonio Carlos Schneider Beck Filho.

1. Heterogeneous systems. 2. Reliability. 3. Near-threshold voltage. 4. Process variation. I. Nazar, Gabriel. II. Schneider Beck Filho, Antonio Carlos. III. Título.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a todos os membros da minha família, meus pais Cláudio e Rejane, e meus irmãos André e Daniela, por todo o apoio oferecido, gratuitamente, ao longo dos meus 100000_2 anos de vida. Em especial, gostaria de agradecer aos meus orientadores, Caco e Gabriel Nazar, pelo apoio, paciência e feedbacks providos ao longo dos últimos sete anos de trabalho, desde o mestrado até o final deste programa de doutorado. É sempre um privilégio trabalhar com pessoas que estão à nossa frente.

ABSTRACT

Technology scaling has been successfully improving the performance of current microprocessors primarily due to the reduced node size that enables increased transistor integration, allowing for the design and widespread adoption of high-performance and highly heterogeneous chips. However, despite the slowdown of Moore's Law, the improved transistor integration is accompanied by complex technological challenges and trade-offs that must be addressed. In particular, smaller technology nodes impose increased reliability, power density, and process variation issues that penalize performance, energy efficiency, and yield. Additionally, overcoming such challenges is especially tricky for devices operating *at the edge* due to the limited power budgets and battery dependency. This thesis, then, proposes a set of methodologies to improve non-functional requirements for heterogeneous chips targeting edge-based applications subject to power, reliability, and process variation constraints. First, we leverage the application and microarchitectural heterogeneity of cores and propose a low-cost learning method for reliability-oriented mappings that provide near-to-optimal Mean Workload to Failure (MWTF) of heterogeneous chips. With the prediction-based mappings, we achieve MWTF as close as 5.6% to the oracle in a low overhead and transparent fashion. Secondly, aiming to improve power-constrained edge devices' performance and energy efficiency, we propose a design-time strategy for chip customization with Near-Threshold Voltage (NTV). Here, we develop an efficient method to allocate NTV and conventional cores in the same die. In this setup, only an optimal subset of the cores are set to operate at NTV, leaving the remaining cores at conventional voltage settings, attenuating the frequency degradation overheads of NTV. Finally, as NTV comes at the cost of exacerbated process variations, we propose a two-step methodology to address delay and power variations on heterogeneous chips. At design time, we augment our chip composition strategy with parameter variation models and develop a statistical and variation-aware design space exploration for heterogeneous chip composition. At the post-design phase, we propose an efficient frequency adaptation mechanism to further cope with unseen parameter variations and improve either performance or yield. We show that under strict power and process variation restrictions, our proposal improves performance by an average of 3.4 times compared to standard NTV approaches and 12% when compared to chips at conventional voltage levels.

Keywords: Heterogeneous systems. Reliability. Near-threshold voltage. Process variation.

Uma Metodologia Visando Melhoria de Confiabilidade e Variação de Processos em Processadores no Domínio da Computação na Borda

RESUMO

A escalabilidade tecnológica tem melhorado com sucesso o desempenho dos microprocessadores atuais, principalmente devido ao tamanho reduzido dos circuitos que permite uma maior integração de transistores, possibilitando o projeto e a adoção generalizada de chips altamente heterogêneos e de alto desempenho. No entanto, apesar da desaceleração da Lei de Moore, a alta integração de transistores é acompanhada por desafios tecnológicos e trade-offs difíceis de serem enfrentados. Em especial, transistores menores impõem problemas de confiabilidade, densidade de potência e variabilidade de processo que penalizam o desempenho, a eficiência energética e o yield quando não são adequadamente abordados. Superar esses desafios é especialmente difícil para dispositivos que operam em ambientes de borda devido aos limites de potência e à dependência de baterias. Nesta tese, propomos uma metodologia abrangente para melhorar os requisitos não funcionais de chips heterogêneos destinados a aplicações de borda sujeitas a restrições de potência, confiabilidade e variabilidade de processo. Primeiro, aproveitamos a heterogeneidade de aplicações e de microarquitetura dos núcleos de processadores e propomos um método de aprendizado de baixo custo para mapeamentos orientados à confiabilidade que fornecem um tempo médio de carga até a falha (MWTF, na sigla em inglês) próximo ao ideal para chips heterogêneos. Com os mapeamentos baseados em previsão, alcançamos um MWTF tão próximo quanto 5,6% do oráculo com baixo custo e de forma transparente. Em segundo lugar, com o objetivo de melhorar o desempenho e a eficiência energética de dispositivos de borda com restrição de potência, propomos uma estratégia de configuração de chips em tempo de projeto com uso de Tensão Próxima do Limiar (NTV). Desenvolvemos uma estratégia eficiente para alocar núcleos tanto NTV quanto convencionais no mesmo chip. Nessa configuração, apenas um subconjunto ótimo dos núcleos opera com NTV, deixando os demais núcleos com configurações convencionais de tensão, reduzindo assim as perdas de frequência decorrentes do uso de NTV. Por fim, como o uso de NTV acarreta variabilidades de processo exacerbadas, propomos uma metodologia em duas etapas para lidar com variabilidades de frequência e potência em chips heterogêneos. No momento do projeto, aprimoramos nossa estratégia anterior de composição de chips com modelos de variabilidade de parâmetros e desenvolvemos uma exploração estatística

e ciente da variabilidade do espaço de design para a composição de chips heterogêneos. Na fase pós-projeto, implementamos um mecanismo eficiente de adaptação de frequência para lidar com variabilidade de parâmetros não previsíveis e melhorar o desempenho ou o yield. Mostramos que, sob restrições estritas de potência e variabilidade de processo, nossa proposta melhora o desempenho, em média, em 3,4 vezes em comparação com abordagens padrão de NTV e em 12% em comparação com chips em níveis convencionais de tensão.

Palavras-chave: Sistemas heterogêneos. Confiabilidade. Tensão próxima ao limiar. Variabilidade de processos.

LIST OF FIGURES

Figure 1.1 Processor trend data up until the year of 2021. 13
Figure 1.2 Thesis outline. The contributions of this thesis are highlighted in red.....18

Figure 2.1 The NMOS transistor....22
Figure 2.2 Potential reduction in the energy required per operation for different supply voltage levels, and the corresponding impact in the transistor delay. 26
Figure 2.3 Frequency spread (right axis) as a function of V_{dd}/V_{th} variation. 27
Figure 2.4 An Alpha (single-ISA) heterogeneous chip....28
Figure 2.5 Evolution of the BOOM processor organization. 32

Figure 3.1 The EnergySmart NTV optimization methodology....38

Figure 4.1 MWTF obtained for two application sets for all possible mappings.....45
Figure 4.2 Assignment graph of application mappings aiming to maximize the overall MWTF. 46
Figure 4.3 The predicted AVF estimation by the neural network compared to the expected values estimated with fault injection....48
Figure 4.4 The predicted AVF estimation by the neural network compared to the expected values estimated with fault injection....52
Figure 4.5 Configuration 1S-1D-2Q 53
Figure 4.6 Configuration 1S-2D-1Q 54
Figure 4.7 Configuration 2S-1D-1Q 54
Figure 4.8 Average MWTF gains for all application sets for the three mapping cases. .55
Figure 4.9 Comparison of the three different heterogeneous configurations (using the predicted mappings) against two homogeneous ones. 56
Figure 4.10 Comparison (average values) against two homogeneous architectures (4D and 4Q). 56
Figure 4.11 MWTF deviation from oracle for different ANN configurations (lower is better)....58

Figure 5.1 Architecture-level view of the chip....59
Figure 5.2 Assignment graph of application mappings aiming to maximize the overall MIPS. 61
Figure 5.3 MIPS distribution for all random workloads when mapped to the heterogeneous chip configurations....65
Figure 5.4 MIPS distribution for the two workloads with the lowest (lowest var) and highest (highest var) degree of MIPS variation across all mappings to heterogeneous chips. 66
Figure 5.5 MIPS comparison for the different MPSoC composition strategies....68
Figure 5.6 Area efficiency ($MIPS/mm^2$) comparison for the different MPSoC composition strategies. 69

Figure 6.1 Power distribution of a heterogeneous chip configuration (4 small + 4 big cores). The samples were generated as explained in section 6.5.1. 70
Figure 6.2 System layers explored in this thesis. 72
Figure 6.3 Abstract workflow of this thesis. 73
Figure 6.4 Architecture-level view of the chip subject to process variations. 74
Figure 6.5 Design time chip exploration workflow. 77
Figure 6.6 The goals of frequency scaling. 80

Figure 6.7 MIPS comparison for the different MPSoC composition strategies.....86
Figure 6.8 MIPS and power samples for optimized chips aiming a power limit of 400mW, with and without FS.....88
Figure 6.9 Average best achievable MIPS for the variation aware (VA-SNAP) and unaware (SNAP) designs, with and without frequency scaling. 88
Figure 6.10 Best attainable MIPS for various yield requirements for the variation-aware VA-SNAP case.....89
Figure 6.11 MIPS performance of all evaluated methods, normalized to the Full NTV (No FS) case.....90
Figure 6.12 Normalized MIPS per each scenario 91

Figure A.1 SmallBoom configuration file (Chisel/Scala code).....106
Figure A.2 Medium configuration file (Chisel/Scala code). 107
Figure A.3 LargeBoom configuration file (Chisel/Scala code).....108
Figure A.4 Per-core attainable MIPS for each scenario explored in Chapter 6 (Tab. 6.2). 109
Figure A.5 Scenario 1. 110
Figure A.6 Scenario 2. 110
Figure A.7 Scenario 3. 110
Figure A.8 Scenario 4. 111
Figure A.9 Scenario 5. 111
Figure A.10 Class diagram of the source code. 112

Figure B.1 Visão geral desta tese. As contribuições principais estão marcadas em vermelho.....114
Figure B.2 Ganhos médios de MWTF para todos os conjuntos de aplicações nos três casos de mapeamento. 115
Figure B.3 Visão de nível de arquitetura do chip. 116
Figure B.4 O mapeamento de aplicações visa maximizar o MIPS total (milhões de instruções por segundo). 117
Figure B.5 Comparação de MIPS para as diferentes estratégias de composição do MPSoC. 117
Figure B.6 Camadas do sistema exploradas nesta tese. 118
Figure B.7 Comparação de MIPS para as diferentes estratégias de composição de MPSoC. 119

LIST OF TABLES

Table 1.1 The optimization requirements tackled in this work with the associated hampering factors, and the proposed (combination) of solutions tackling each constraint. 15

Table 2.1 Some examples of popular commercial heterogeneous MPSoCs from three leading smartphone companies. 29

Table 3.1 Previews works on addressing varying optimization goals. 40

Table 4.1 Explored core configurations. 44
Table 4.2 Area (mm^2) and approximate normalized flip-flop raw SER (based on the number of flip-flops) of the explored chips 51
Table 4.3 Percentage of application sets that maximize a given metric when prediction-based mapping is applied to each evaluated chip configuration. Example: 62.8% of the application sets achieve the highest MWTF with configuration 1S-2D1-Q. 58

Table 5.1 Explored core configurations. The power and frequency shown are under nominal conditions. 63
Table 5.2 Suite of edge-applicable tasks adopted in this work. 64
Table 5.3 The three evaluated chip cases. Four NTV rocket cores (**4R**) are considered in Config 3. All other cores are at STV. 65
Table 5.4 Mapping distribution characterization, for each configuration, for the two workloads with the lowest and highest variation across mappings 67

Table 6.1 Explored processor configurations. The power and frequency shown are under nominal conditions 84
Table 6.2 Application scenarios explored in this work. 85
Table 6.3 Best average achievable MIPS for varying yield requirements (average across all power budgets). Gains are the geometric mean of all gains, across all budgets, provided by frequency scaling. 87
Table 6.4 MIPS gains of VA-SNAP (+FS), per scenario, against all other chip design cases. 91

LIST OF ABBREVIATIONS AND ACRONYMS

ANN	Artificial Neural Network
AVF	Architectural Vulnerability Factor
DMR	Dual Modular Redundancy
DSE	Design Space Exploration
DSP	Digital Signal Processor
DVFS	Dynamic Voltage and Frequency Scaling
ECC	Error Correcting Code
GPU	Graphics Processing Unit
ILP	Instruction-Level Parallelism
ISA	Instruction Set Architecture
MPSoC	Multiprocessor System on a Chip
MWTF	Mean Workload to Failure
NTC	Near-Threshold Computing
NTV	Near-Threshold Voltage
RMT	Redundant Multi-Threading
RTL	Register-Transfer Level
SDC	Silent Data Corruption
SER	Soft Error Rate
SEU	Single-Event Upset
STC	Super-Threshold Computing
STV	Super-Threshold Voltage
TDP	Thermal Design Power
TMR	Triple Modular Redundancy
VLSI	Very Large-Scale Integration

CONTENTS

1 INTRODUCTION 13
1.1 Challenges Addressed in this Thesis 14
1.2 Contributions of this Thesis 17
1.2.1 Contribution 1: Reliability-Oriented Mapping Solutions 17
1.2.2 Contribution 2: Efficient use of NTV for Improved Performance and Energy Efficiency 19
1.2.3 Contribution 3: Addressing Process Variations 19
1.3 Structure of this Document 20
2 BACKGROUND 21
2.1 Sources of Power Consumption and Optimization Methods 21
2.2 Radiation-Induced Soft Errors and Fault Tolerance Concepts 23
2.3 Near-Threshold Voltage Computing 25
2.3.1 Timing Failures 28
2.4 State-of-the-Art Heterogeneous Architectures 28
2.5 The Chipyard Framework 30
2.5.1 The BOOM and Rocket Cores 31
3 RELATED WORK 34
3.1 Related Works on Addressing Soft Errors Reliability 34
3.1.1 Redundancy-based Fault Tolerance Approaches 34
3.1.2 Mapping-based Fault Tolerance Approaches 35
3.2 Related Works on Process Variation Mitigation 37
3.3 Contextualizing this Thesis with Respect to Previous Works 40
4 APPLICATION MAPPING APPROACHES TO IMPROVE RELIABILITY AND PERFORMANCE OF HETEROGENEOUS SYSTEMS 42
4.1 Improving MWTF with Application Mapping 42
4.1.1 Motivation and Background 44
4.2 Adaptive Mapping 46
4.2.1 Problem Definition 46
4.3 Experimental Methodology 50
4.4 Results on Reliability-Oriented Mappings 53
4.4.1 AVF Prediction 53
4.4.2 Dynamic Mapping Evaluation 55
4.4.3 Comparing Heterogeneous versus Homogeneous Configurations 56
4.4.4 Distribution of best configurations 57
4.4.5 Exploiting Different ANNs 58
5 A POWER-EFFICIENT AND PERFORMANCE-ORIENTED EXPLORATION METHODOLOGY WITH NTV CHIPS 59
5.1 Introduction 59
5.2 Chip Architecture Exploration Scope 60
5.3 Application Mapping with Heterogeneous Systems 60
5.4 Architectural Search and Optimization Goal 61
5.5 Results on Performance-Oriented Mappings 63
5.5.1 Experimental Methodology 63
5.5.2 Results 65
5.5.2.1 Evaluating Application Mappings on Heterogeneous Cores 65
5.5.2.2 Performance Evaluation of SNAP 67
5.5.2.3 Area Efficiency Evaluation 69

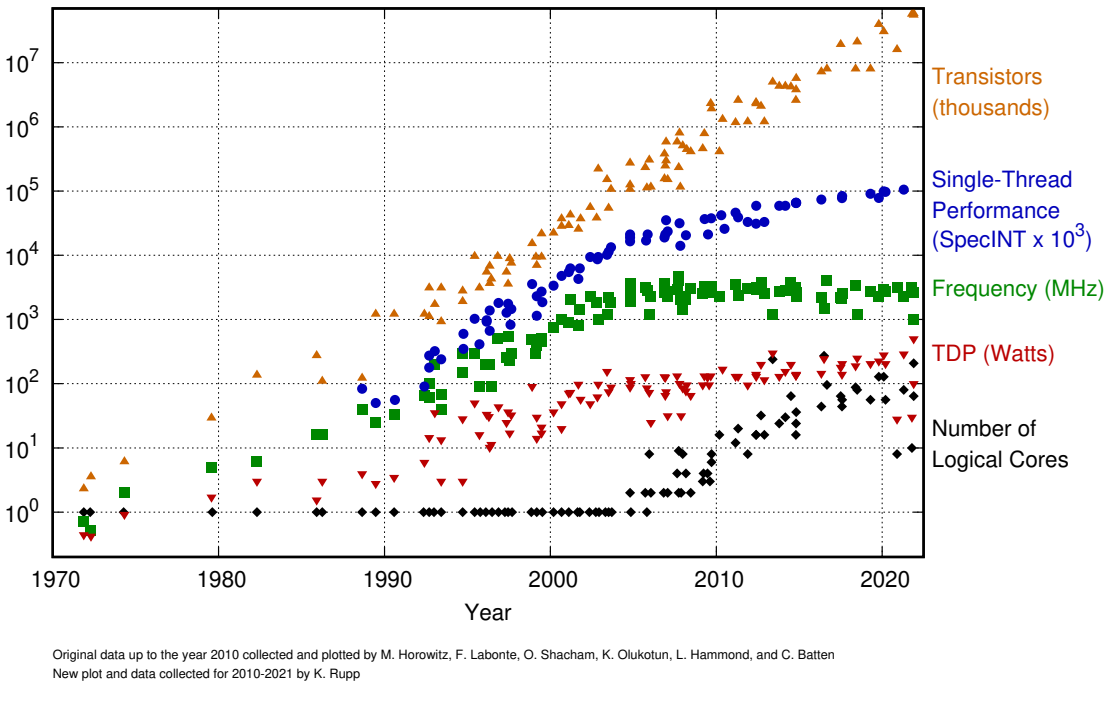
6 A VARIATION-AWARE METHODOLOGY FOR IMPROVED PROCESSOR DESIGNS FOR THE EDGE COMPUTING DOMAIN 70
6.1 Motivational Analysis 70
6.2 Variation-Aware Proposal 71
6.3 Approach Overview 73
6.3.1 Proposed Flow of Optimization 73
6.3.2 Chip Architecture Exploration Scope 73
6.3.3 Modeling and Addressing Parameter Variations 74
6.4 Variation-Aware Design- and Post-Design time Optimization 77
6.4.1 Design-time and Variation-Aware Chip Customization 77
6.4.2 Post-Design and Variation-Aware Frequency Scaling 80
6.4.3 Putting It All Together: Variation-Aware Exploration Algorithm 82
6.5 The Proposal's Evaluation 83
6.5.1 Experimental Methodology 83
6.5.2 Results 86
6.5.2.1 Variation-Aware VA-SNAP Approach and Frequency Scaling 86
6.5.2.2 Case Study 87
6.5.2.3 Variation-aware versus Variation-unaware Approaches 89
6.5.2.4 MIPS for Minimum Yield Requirements 89
6.5.2.5 Overall Gains Evaluation 90
6.5.2.6 Per-Scenario Results 91
7 CONCLUSIONS 92
7.1 Addressing Reliability 92
7.2 Improving Performance and Energy Efficiency with NTV Edge Devices 93
7.3 Addressing Process Variations with NTV Edge Devices 93
7.4 Future Work 94
7.5 List of Published Papers 95
7.5.1 Main Publications 96
7.5.2 Publications as a Collaborator 96
REFERENCES 97
APPENDIX A — IMPLEMENTATION DETAILS AND PER-SCENARIO EVALUATION 106
A.1 Detailed Configurations of the Explored Cores 106
A.2 Per-Scenario VA-SNAP Evaluation 109
A.3 Class Diagram of the Architecture 112
APPENDIX B — UMA METODOLOGIA VISANDO O APRIMORAMENTO DE PROCESSADORES RESTRITOS À VARIAÇÃO DE PROCESSOS E APLICÁVEIS À COMPUTAÇÃO DE BORDA 113
B.1 Parte 1: Otimização de Confiabilidade 115
B.2 Parte 2: Otimização de Desempenho e Eficiência Energética 116
B.3 Parte 3: Abordando Variabilidade de Processos 118

1 INTRODUCTION

The progressive improvements in technology scaling and the associated increased transistor integration have resulted in the emergence of a myriad of heterogeneous chip designs to accommodate the performance demands of a diverse domain of complex applications. Such integration trends are shown in Fig. 1.1, which characterizes the evolution of microprocessors up until the year 2021 (data obtained from (RUPP, 2021)). Despite the slowdown of Moore's Law, there has been continuous and exponential integration of transistors over the decades. On the other hand, no significant frequency improvements have been achieved since around 2005, which resulted in roughly the stagnation of single-threaded performance and, as a consequence, the emergence of the multicore paradigm and the urge for more hardware specialization in the form of domain-specific architectures (HENNESSY; PATTERSON, 2019).

Specially, such technological progress has enabled the emergence of heterogeneous multicores applicable to the *edge computing* domain (Shi et al., 2016; Wu et al., 2019). In this computing environment, applications are executed by edge devices that operate near the network's edge. In general, edge devices can be any form of processing engine between the end user and the cloud (e.g., a device operating near a data sensor

Figure 1.1 – Processor trend data up until the year of 2021.



Source: (RUPP, 2021)

that provides the input data to the device). Executing applications (or parts of them) at the edge is compelling for latency-critical operations since it allows for improved performance and even energy efficiency since edge-based execution avoids offloading input data to remote servers that usually rely on unpredictable network latency, mainly due to uncertain server-side performance caused by CPU load variance and queuing (Wang et al., 2020; LI; ZHOU; CHEN, 2018; KANG et al., 2017).

In addition to the hardware-side amelioration of commodity microprocessors, the growing commercial interests of both hardware and software industries have resulted in the proliferation of many newer and more performance-demanding applications. Usually, applications tend to be considerably heterogeneous due to the high number of different domains they span. For example, up to July 2023, Google Play alone has over 2.6 million published Android applications spanning across several distinct categories such as entertainment, education, business, online shopping, and many others (STATISTICS, 2023). This further paves the way for the adoption of *heterogeneous systems*, where state-of-the-art power/performance optimization strategies often harness from *software heterogeneity* by adopting *hardware heterogeneity* across different layers of the system (e.g., from circuit- to system-level), in which the combination of both hardware and software heterogeneity can be explored to improve key metrics (e.g., performance, power, energy, reliability, etc.) by leveraging application-to-core affinities and mapping applications to the most suitable core.

However, despite the progressive improvements in computing systems, many edge-side optimization challenges are still associated with transistor downscaling. Among others, some important and challenging constraints are (1) radiation-induced soft errors, (2) the chip-specific power envelope and the associated power wall issues, and (3) transistor-level process variation. When not addressed, such constraints hinder many (functional and non-functional) essential requirements of microprocessors, such as reliability, performance, energy efficiency, and yield. This thesis, then, aims at optimizing such requirements of heterogeneous chips subject to the three constraints mentioned above.

1.1 Challenges Addressed in this Thesis

We illustrate the optimization requirements addressed in this thesis, their associated hampering factors, and the proposed solutions in Tab. 1.1. We justify addressing the considered hampering factors in the domain of heterogeneous multicores as follows:

Table 1.1 – The optimization requirements tackled in this work with the associated hampering factors, and the proposed (combination) of solutions tackling each constraint.

Requirement	Hampering factors	Proposed solutions
Reliability	Single-event upsets	Reliability-oriented application mapping
Energy efficiency	Power wall	Efficient use of NTV + Frequency adaptation
	Process variation	DSE with heterogeneous cores + Frequency adaptation
Performance	Power wall	Efficient use of NTV + Frequency adaptation
	Process variation	DSE with heterogeneous cores + Frequency adaptation
	Domain-specific power limits	DSE with heterogeneous cores + Efficient use of NTV + Application mapping + Frequency adaptation
Yield	Process variation	DSE with heterogeneous cores + Frequency adaptation

Radiation-induced faults. The diminishing sizes of transistors and the associated reduced voltage of contemporary microprocessors historically pose reliability challenges that arise from radiation-induced faults - frequently in the form of Single-Event Upsets (or SEUs) - that can cause malfunction in mission-critical applications (MITRA et al., 2014; HENKEL et al., 2013). Additionally, while past reliability research has focused mostly on single cores or homogeneous multicore systems, the limitations in Instruction-Level Parallelism (ILP) have led to the adoption of heterogeneous multicore architectures, such as the big.LITTLE and DynamIQ (ARM, 2021) architectures to meet the demands for more task-level throughput. *We make the case that reliability research on heterogeneous multicores is mostly an unexplored topic, and we provide an efficient and low-cost reliability-oriented application mapping optimization methodology for this domain.*

Power Wall. The current transistor scaling approaches do not provide simultaneous improvements in feature size and voltage (V_{dd}) because the threshold voltage has not been scaled at the same pace as the transistor's dimension to keep the leakage current under control (Borkar et al., 2003; Horowitz et al., 2005; Bohr, 2007). Hence, V_{dd} does not scale linearly across different technology generations as it has been kept roughly constant for over one decade and, consequently, power density tends to increase for smaller tech-

nology nodes (Bohr, 2007). This trend, then, introduces the phenomenon known as *the end of Dennard's scaling* (Esmaeilzadeh et al., 2011), which claims that voltage should scale linearly with transistors' dimensions and, consequently, power density would stay roughly constant over the generations (Dennard et al., 1974). This contrasts with the actual observed trend of increased power density. In essence, this *power wall* phenomenon implies that frequency cannot be further improved reliably due to the stagnation of V_{dd}, leading single-core performance to stagnate. One approach to mitigate the power wall issues is the adoption of low voltage designs based on Near-Threshold Voltage (NTV) (Dreslinski et al., 2010), which consists in setting V_{dd} to a point close to the threshold voltage. While this comes at the cost of frequency degradation and exacerbated process variations, NTV can improve energy efficiency and throughput under a power envelope when used smartly. *We adopt power-aware and efficient use of NTV, reliably at low frequency, to mitigate power wall issues.*

Domain-specific power limits. In addition to the power wall issues imposed by circuit-level constraints, executing compute-demanding applications on edge devices is challenging to achieve at low power limits. Usually, system-level power envelopes are imposed by Thermal Design Power (TDP) constraints (e.g., due to the limited cooling capabilities) or battery dependence, which demands high energy efficiency. Thus, more aggressive solutions for multicore optimization may be necessary to accommodate such restrictive requirements if high performance is needed for this domain. *We propose an efficient Design Space Exploration (DSE) methodology in both the microarchitectural and voltage settings layer and propose a special NTV optimization method to improve performance with minimal power slacks under a power limit. Although NTV degrades frequency, our strategy reclaims performance by efficiently increasing the number of cores under a power envelope.*

Process variation. Unfortunately, NTV optimization comes at the cost of exacerbated process variations (Kaul et al., 2012; Karpuzcu et al., 2012) that must be tackled before chip deployment. Process variations are raised by fabrication effects (e.g., due to lithography imperfections), which lead to delay and static power deviations from the intended design goals (KARPUZCU; KIM; TORRELLAS, 2013; Kaul et al., 2012; PINCKNEY et al., 2012). If no countermeasures are taken, such unwanted parameter fluctuations can hinder performance (e.g., if variation affects the critical path, effectively hampering achievable frequency), power, and energy efficiency (PINCKNEY et al., 2012). Specially, variation-induced power overheads beyond the required power limits can degrade yield,

as a fraction of the assembled chips will not comply with the application requirements, possibly having to be discarded due to excessive leakage (Borkar et al., 2003). *We augment our DSE methodology with process variation models and propose a combination of design-time and post-design-time approaches to provide chip designs to mitigate the discussed variation issues.*

1.2 Contributions of this Thesis

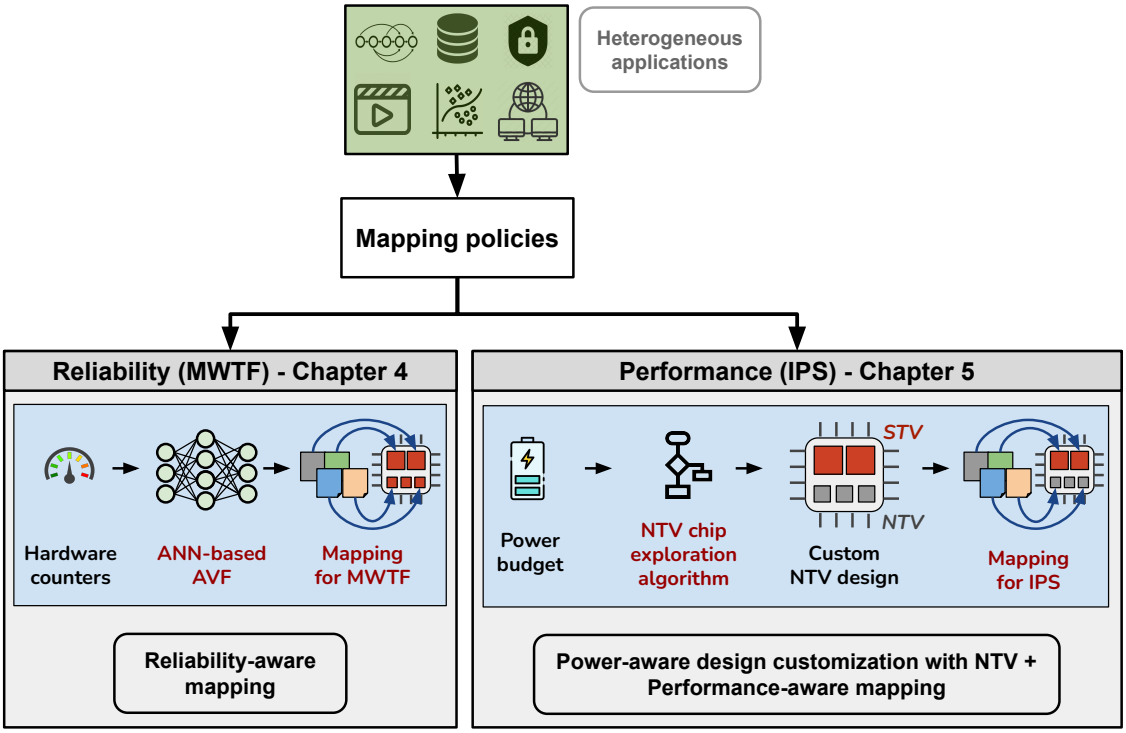
An outline of this thesis's workflow and contributions is shown in Fig. 1.2. The thesis is divided into three phases addressing different concerns in the scope of heterogeneous multicores for the edge domain. First, in Chapter 4, we propose a transparent mapping solution aiming at improving reliability in the context of heterogeneous multicores. Secondly, in Chapter 5, we propose a design methodology to improve performance and energy efficiency, under strict power limits, of heterogeneous multicores applicable to low-power edge scenarios. Finally, in Chapter 6, we augment our chip design methodology to address process variations and propose a two-step approach (design- and post-design time) that provides customized and more efficient chips that improve performance, energy efficiency, and yield under a power envelope and process variation contexts. The contributions are described in the following sections.

1.2.1 Contribution 1: Reliability-Oriented Mapping Solutions

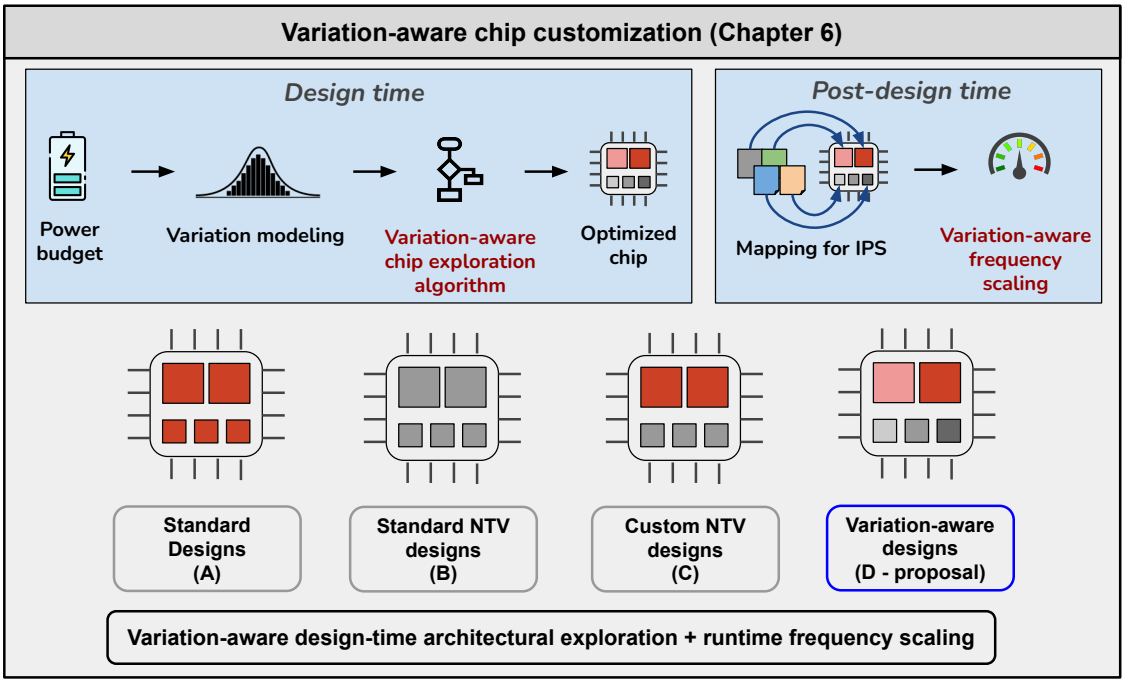
In this part, we leverage core-level heterogeneity from microarchitectural resources to perform proper workload mappings to the most suitable core type to maximize reliability. This is shown in Fig. 1.2a (left block). Our proposal consists in adopting a learning-based and automatic solution for efficient mappings to maximize the reliability of the system in terms of Mean Workload to Failure (MWTF) on heterogeneous multicores. Here, we perform runtime monitoring of application-dependent core pipeline utilization with hardware counters and train an Artificial Neural Network (ANN) to estimate the core's Architectural Vulnerability Factors (or AVF) from the hardware counters' data. We then perform near-to-optimal AVF-oriented application mappings to heterogeneous cores aiming at maximizing the chip-level system's MWTF;

Figure 1.2 – Thesis outline. The contributions of this thesis are highlighted in red.

(a) In the first part we explore mapping policies aiming at maximizing reliability and performance.



(b) The second part aims to compose energy efficient and variation-aware chip designs with NTV.



1.2.2 Contribution 2: Efficient use of NTV for Improved Performance and Energy Efficiency

In the second part, in Fig. 1.2a (right block), we propose an efficient DSE methodology for composing micro-architecturally heterogeneous multicores under a domain-specific power envelope. We additionally propose a method for selective NTV, where only an efficiently selected set of the cores that compose the chip operate at NTV. This results in a chip comprising two voltage islands, accommodating both NTV and conventional core designs in the same die, with improved energy efficiency and performance under a power limit. At runtime, we combine the chip customization method with runtime application mappings to increase task-level instruction throughput. This renders chips with maximized Instructions Per Second (IPS) and minimal power slacks under the power envelope.

1.2.3 Contribution 3: Addressing Process Variations

In this part of the thesis, we augment our DSE methodology for partial NTV and propose a variation-aware chip exploration methodology to compose power-constrained chips subject to process variation scenarios. This part is shown in Fig. 1.2b. First, we propose a sampling-based and variation-aware strategy during the design stage to select optimized chip configurations with variation-aware and selective NTV. Secondly, we propose a frequency adaptation mechanism in the post-design time to further cope with variations. Combining both design- and post-design steps provides IPS-optimized and power-aware configurations while maintaining minimum yield requirements.

Our procedure relies on exploring heterogeneous multicore designs with varying microarchitecture cores. For example, a combination of performance-oriented cores plus low-power cores results in improved chip-level energy efficiency. Four abstract heterogeneous chip design cases are shown in Fig. 1.2b (the A-D designs, bottom part). The designs are (**A**) standard designs that rely on a single (conventional) and safe voltage setting suitable for frequencies at the Giga-Hertz level; (**B**) conventional NTV designs that allocate a single NTV voltage level to all cores in the chip, at low frequencies at the Mega-Hertz level; (**C**) custom architectures that efficiently combines both STV and NTV cores in the same chip; and (**D**), our proposed variation-aware chip composed of STV and NTV cores.

This contribution consists of providing customized chip configurations following the **D** paradigm, as it is suitable to improve performance and energy efficiency at low-power scenarios while maintaining minimum yield requirements under process variations constraints.

1.3 Structure of this Document

This work is structured as follows:

- Chapter 2: Provides background information on microprocessor reliability, Near-Threshold Computing, and its associated process variation challenges. We also briefly discuss the emergence and importance of heterogeneous systems and the toolchain adopted to develop this work;
- Chapter 3: Highlights previous works on microprocessor reliability improvement proposals, as well as previous approaches to mitigate process variations in the context of NTV and conventional designs;
- Chapter 4: Details our proposal for improved reliability on heterogeneous systems;
- Chapter 5: Details our NTV proposal for improved performance and energy efficiency of low-power heterogeneous systems applicable to the edge;
- Chapter 6: Details our approach for variation-aware and low-power chip designs applicable to the edge;
- Chapter 7: Highlights the main conclusions of the chapters 4, 5, and 6;
- Appendix A: Provides details on the explored core configurations and the evaluated application scenarios;
- Appendix B: Summarizes this work in Portuguese (required).

2 BACKGROUND

This chapter provides background information on key concepts such as basics on transistor behavior and sources of power dissipation (Sec. 2.1), NTV and the associated process variations and reliability issues (Sec. 2.3), current state-of-the-art heterogeneous systems (Sec. 2.4) as well as a brief discussion on an open source framework (*Chipyard*) we adopted to explore heterogeneous systems design (Sec. 2.5).

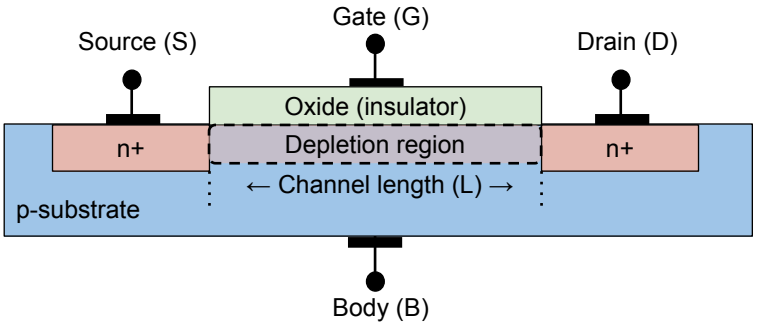
2.1 Sources of Power Consumption and Optimization Methods

The Complementary Metal Oxide Semiconductor (CMOS) technology is the building block for the implementation of digital logic. The CMOS logic relies on a specific arrangement of Metal-Oxide-Semiconductor Field Effect Transistor (MOSFET) to construct logic gates. Such transistors are considered the most widely manufactured device in history (HANDY, 2014) mainly due to their low power consumption and scalability properties. As an example, Fig. 2.1 depicts the high-level view of an NMOS transistor, which consists of three main terminals: source (S), gate (G), and drain (D). Different modes of operation are achieved depending on the relationship between the voltages applied to the terminals and the threshold voltage (V_{th}). Three regions of operation are possible:

- Sub-threshold region: when $V_{gs} < V_{th}$ and hence there is no channel formed between source and drain. Even though the transistor is considered off in this region, there is drain-to-source *leakage* current (I_{ds}) flowing according to Eq. 2.1 (RABAEY, 1995).
- Linear region: when $V_{gs} > V_{th}$ and $V_{ds} < V_{gs} - V_{th}$. A channel is formed between the source and drain (depletion region). In this region, the I_{ds} current depends on the drain voltage, and the transistor behaves like a resistor that can be controlled by V_{gs}.
- Saturation region: when $V_{gs} > V_{th}$ and $V_{ds} > V_{gs} - V_{th}$. A channel is formed between the source and drain (depletion region), but the I_{ds} current saturates and does not depend on the drain voltage.

Notice in Eq. 2.1 that in the sub-threshold region, the leakage current depends exponentially on V_{ds}, V_{gs}, V_{th} and the thermal voltage (V_T). The leakage current is also sensitive to transistors' parameters, such as the gate length (L), width (W), and tempera-

Figure 2.1 – The NMOS transistor



ture.

$$I_{ds} = I_0 \frac{W}{L} e^{\frac{V_{gs}-V_{th}}{nV_T}} (1 - e^{-\frac{V_{ds}}{V_T}}), V_T \propto Temp \tag{2.1}$$

The gate delay can be modeled in high-level as shown in Eq. 2.2 (RABAEY, 1995), which is the time taken for a signal to propagate from the gate input to its output. For a CMOS inverter, for example, the delay is defined as the time the output takes to reach $V_{dd}/2$ (low to high - LH) after the input reaches $V_{dd}/2$ (high to low - HL).

$$t_{p,LH} \approx kC \frac{V_{dd}}{I_{ds}^{PMOS}}, t_{p,HL} \approx kC \frac{V_{dd}}{I_{ds}^{NMOS}} \tag{2.2}$$

In the equation, k is a fitting parameter, and C is the gate output capacitance. The gate delay determines the maximum frequency (f) for the gate.

Logic gates dissipate static and dynamic power. Dynamic power dissipation is caused by input transitions that lead the gate outputs to toggle, and is calculated as $P_{dyn} = Cf\alpha V_{dd}^2$, where α is the application-dependent switching activity (i.e., probability of state transition). Static power happens due to leakage current, which can be expressed as $P_{leak} = I_{leak} V_{dd}$, where I_{leak} is determined by Eq. 2.1. The total power consumption for generic gates can then be expressed as $P = P_{dyn} + P_{leak}$, which influences the energy and energy efficiency of the circuit for a given performance target.

Optimizing power consumption is essential to increase both energy and thermal efficiency, which is relevant for cloud servers, desktop computers, and edge devices. However, keeping low power consumption for edge devices is crucial due to 1) battery dependence (needs energy efficiency) and 2) restricted temperature (needs thermal efficiency), which raises cooling challenges for portable devices, for instance. Over the years, many low-power and/or energy efficiency optimization strategies have been proposed across different layers, such as clock or power gating (PEDRAM, 1996), Dynamic

Voltage and Frequency Scaling (DVFS) (BURD et al., 2000; SEMERARO et al., 2002), approximate computing at various distinct layers (MITTAL, 2016), aggressive voltage underscaling with NTV (Kaul et al., 2012), and system-level strategies such as the adoption of application-specific accelerators and heterogeneous cores such as the ARM's DynamIQ (ARM, 2021).

In this work, we explore optimization strategies by adopting NTV applied to heterogeneous systems. We justify the adoption of NTV due to its increased energy efficiency improvements and also as an aggressive power capping method. When compared to conventional STV designs, NTV provides improved energy efficiency as the most energy-efficient voltage point is known to be close to the transistor's threshold voltage (Dreslinski et al., 2010; Markovic et al., 2010); thus, NTV provides more aggressive power reduction and increased energy efficiency when compared to standard DVFS approaches at conventional voltage levels, e.g., (Dighe et al., 2011; RAGHUNATHAN et al., 2013; Teodorescu; Torrellas, 2008), mostly because NTV is suited to reduce both dynamic and static power dissipation.

Moreover, given the widespread and firm adoption of heterogeneous systems in commercial devices, we also consider NTV in conjunction with proper application mapping to better exploit the heterogeneity of cores and applications so that better application-to-core affinities are exploited to increase performance further. The next section briefly introduces NTV concepts and the associated design challenges.

2.2 Radiation-Induced Soft Errors and Fault Tolerance Concepts

Radiation-induced soft errors, mostly in the form of Single-Event Upset (SEU), may occur when energized particles (e.g., from cosmic rays or alpha particles from decaying materials) hit a transistor and flips the value of the stored bit (flip-flops or SRAM cells) (MITRA et al., 2014; HENKEL et al., 2013). The soft error rate is strongly dependent on the transistor's critical charge (the minimum charge required to invert the bit state), which in turn depends on V_{dd} (JAHINUZZAMAN; SHARIFKHANI; SACHDEV, 2009).

To measure the susceptibility of application failures due to soft errors, the authors in (MUKHERJEE et al., 2003) have introduced the concept of Architectural Vulnerability Factor (AVF) as the conditional probability of system failure given that a bit-flip occurred in the microprocessor. The AVF depends on the fraction of bits in the processor that are

required for the correct execution of the application, called Architectural Correct Execution bits (or ACE bits). The fraction of ACE bits is directly related to the degree of useful *occupancy* of each internal structure in the processor's pipeline, which is determined by complex correlations between several microarchitectural factors and applications' characteristics that can change the residence times of useful bits in the structures (MUKHERJEE et al., 2003; WALCOTT; HUMPHREYS; GURUMURTHI, 2007). For instance, a microprocessor with good branch prediction accuracy and too much memory stalls due to cache misses will tend to fill up the processor's internal structures with useful data, increasing the occupancy (or degree of utilization) of the structures and consequently making it more vulnerable to bit-flips.

Standard resiliency analysis to investigate fault tolerance levels requires measuring the processor's AVF, which is determined by estimating the fraction of ACE bits in the processor's pipeline for distinct applications. A common approach to identify the fraction of ACE bits in processors is termed *ACE analysis*, which consists in monitoring the application's instruction trace and gathering instruction-level metrics (such as ISA register utilization) with high-level simulators. This method, however, is well-known for providing overly pessimistic vulnerability factors (WANG; MAHESRI; PATEL, 2007; GEORGE et al., 2010). In order to avoid the limitations of ACE analysis or the adoption of high-level simulations that do not provide hardware details, we measure the AVF through fault injections at RTL, gathering more realistic (and less conservative) AVF estimations.

The raw Soft Error Rate (SER) of a system is the rate of raw/total soft errors experienced by the chip (e.g., how many bits are flipped per unit of time in the circuit), and it is commonly expressed as the raw FIT rate (Failures in Time) - the number of raw errors experienced in 1 billion hours of operation. This error rate depends on manufacturing parameters (e.g., transistor sensitivity due to the critical charge and voltage of operation), environmental conditions (e.g., radiation due to proximity to alpha-decaying materials or space applications exposed to high levels of cosmic rays), and silicon area exposed to soft errors. Therefore, SER is proportional to the core's area and the circuit technology. Notice that the raw SER does not determine the frequency of errors that lead to actual failures, as most of such errors are masked at different layers such as circuit-level, microarchitectural-level, or even application-level masking factors. The *effective* SER is the rate of errors that actually lead to system failures, which is measured by *derating* the raw SER by the AVF/derating factor ($SER_{effective} = SER_{raw} \times AVF$).

While the AVF is a satisfactory metric for estimating the probability of failures in

the presence of bit-flips in the processor, more than such a metric is needed to evaluate the fault tolerance of heterogeneous systems in which application mapping strategies influence both AVF and performance. For example, larger cores may execute the workload faster, but they also have higher SER (due to larger area) and application-dependent AVF. The AVF metric does not capture all information on the effects of resilience across different application mappings because mappings affect both AVF and performance. Hence, only considering the AVF as a vulnerability metric in the context of heterogeneous cores with varying performance will likely be misleading in this scenario. To account for that, the resilience metric we consider in this work is the Mean Workload to Failure (MWTF) (REIS et al., 2005), defined in Eq. 2.3, because it accounts for both the AVF and workload execution time, which are affected by application mapping. Higher MWTF means more computation is completed before the next system failure, which can be achieved by adopting proper mapping solutions to heterogeneous cores.

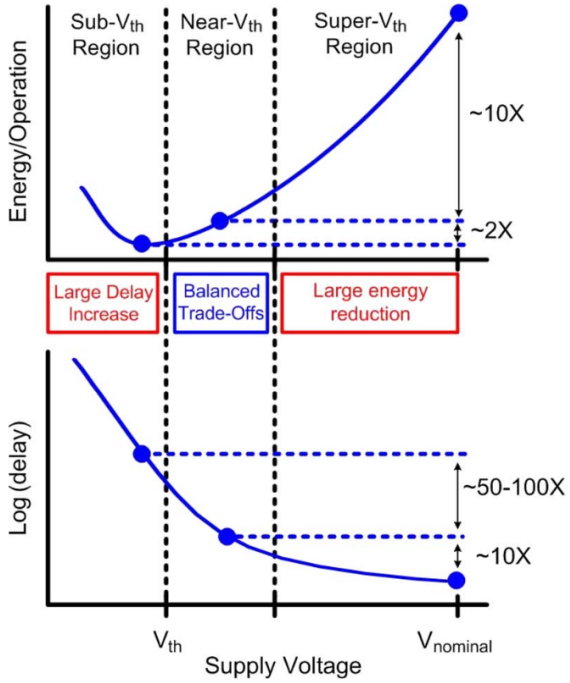
$$\begin{aligned} MWTF &= \frac{amount\ of\ workloads\ computed}{number\ of\ errors\ encountered} \\ &= (SER_{raw} \times AVF \times execution\ time)^{-1} \end{aligned} \tag{2.3}$$

2.3 Near-Threshold Voltage Computing

Near-Threshold Voltage (NTV), in contrast to the conventional Super-Threshold Voltage (STV), consists in operating the circuit at a voltage very close to the threshold voltage level (V_{th}) to achieve aggressive power reduction at the cost of frequency degradation (Dreslinski et al., 2010; Chang et al., 2010; Markovic et al., 2010). NTV provides reduced dynamic power dissipation due to the quadratic dependence on V_{dd} and reduced static power dissipation (linearly or exponentially, depending on the transistors' region of operation).

Ideally, the NTV power reduction is aggressive enough to cover the frequency degradation, which is usually achieved by increasing the chip-level parallelism under a power envelope. For instance, by increasing the number of cores, or processing elements in general, NTV designs provide improved energy efficiency, as shown in Fig. 2.2. However, the energy efficiency gains stop at some point near the sub-threshold region because at this point the drain-to-source current depends exponentially on V_{dd}, which brings about orders of magnitude of frequency degradation due to the increased transistor delay.

Figure 2.2 – Potential reduction in the energy required per operation for different supply voltage levels, and the corresponding impact in the transistor delay.



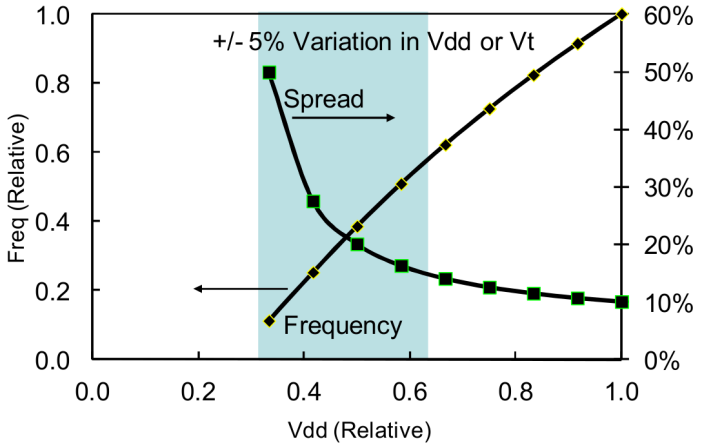
Source: (Dreslinski et al., 2010)

Unfortunately, a secondary and unwanted effect of NTV is the increased susceptibility to parametric variations (i.e., deviations from the nominal values) that arises due to Process Variations (PV). PV is already a known issue at STV (Bowman; Duvall; Meindl, 2002; Fu; Li; Fortes, 2009; Dighe et al., 2011). At NTV, however, PV effects become more pronounced, up to 20 times higher when compared to conventional STV regime (Markovic et al., 2010).

Process variations (die-to-die and within-die variations) manifest mostly through deviations in the transistors' V_{th} and effective channel length (L_{eff}), which are mostly attributed to design-time effects such as *systematic effects* (due to lithographic imperfections) and *random effects* (due to variable doping concentrations). At runtime, PV can also be affected by aging, voltage, and temperature (Bowman; Duvall; Meindl, 2002; Fu; Li; Fortes, 2009; Karpuzcu et al., 2012; Karpuzcu et al., 2013). Consequently, parametric variations affect both transistor's delay and leakage current, possibly shifting the optimal voltage point, the voltage point that yields the best energy efficiency, to a value higher than the ideal/PV-free circuit (PINCKNEY et al., 2012).

For a circuit operating at NTV, a small variation in the supply voltage results in a large change in the transistor delay, hence influencing the frequency of operation. Fig. 2.3 depicts the relative frequency of operation and frequency variation (measured by dividing

Figure 2.3 – Frequency spread (right axis) as a function of V_{dd}/V_{th} variation.



Source: (Kaul et al., 2012)

the standard deviation of frequency by the average frequency of sampled circuits in a die, or σ/μ) as a function of supply voltage. Near the NTV region, even a small variation in supply voltage or V_{th} may result in up to 50%

To further elucidate why variation is accentuated at NTV, an accurate model for current and delay near the threshold region is necessary. For that, current and delay formulas for the NTV region are derived according to the *EKV model* (ENZ; KRUMMENACHER; VITTOZ, 1995; Markovic et al., 2010), according to the Eq. 2.4 and 2.5, respectively, which cover all regions of operation.

$$I = \frac{\mu}{L_{eff}} \times n \times vt^2 \times ln^2(e^{\frac{Vgs-V_{th}}{2\times n\times vt}} + 1) \tag{2.4}$$

$$delay_{gate} \propto \frac{V_{dd} \times L_{eff}}{\mu \times n \times vt^2 \times ln^2(e^{\frac{Vgs-V_{th}}{2\times n\times vt}} + 1)} \tag{2.5}$$

In the equations above, μ represents the carrier mobility, vt is the thermal voltage, and n is a process-dependent parameter that relies on sub-threshold characteristics.

The equations show the strong dependence of the gate delay on both V_{th} and L_{eff}. For a multicore setting with variations in such parameters, for example, if a given design decision relies on the slowest core to determine the frequency of operation, which is a conventional design strategy while not addressing the variation issues, then the core-to-core delay variations translate into different V_{opt} points for different cores. Therefore, corner-based optimizations, in which the slowest core determines the frequency, yield a sub-optimal energy efficiency solution (Zhai et al., 2007; Karpuzcu et al., 2013) as different cores have different optimal voltage settings. We elucidate other approaches to

address process variation in Sec. 3.2.

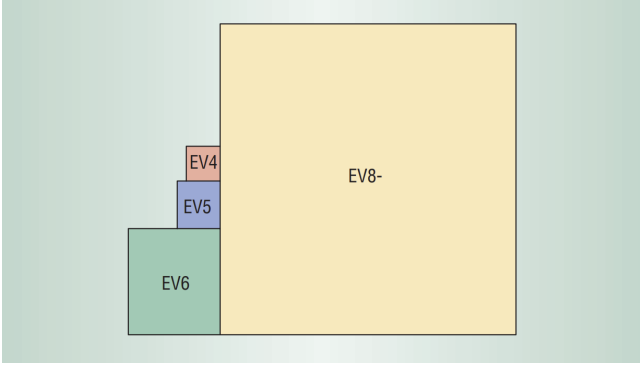
2.3.1 Timing Failures

Timing failures occur in both combinational and sequential circuits. If a combinational path is too slow to process the inputs for the designed clock frequency (due to unaddressed variation issues), timing failures may arise in the output of the circuit due to timing violations (i.e., the frequency of operation is too high for the combinational (variation-afflicted) path) (Ernst et al., 2004). Secondly, sequential circuits with conventional 6T SRAMs cells, for example, variations can cause both read, write and hold failures due to the tight timing margins of operation because the access transistors of such cells impose conflicting timing requirements (for read and write operations) as such cells are designed to be small to achieve high bit density (Karpuzcu et al., 2012).

2.4 State-of-the-Art Heterogeneous Architectures

The need to improve compute performance under restricted power budgets has raised many architectural challenges, and the emergence of heterogeneous systems is one leading way to address such issues. Single-ISA heterogeneous (or asymmetric) systems are any form of multi-core MPSoC composed of cores with varying microarchitectural properties, but all implementing a unique Instruction Set Architecture (ISA) specification. Fig. 2.4 depicts a high-level view of a multi-core processor comprising different cores implementing the Alpha ISA. Such architectures were first proposed aiming at reduced power consumption (KUMAR et al., 2003), and later to improve multi-threaded or

Figure 2.4 – An Alpha (single-ISA) heterogeneous chip.



Source: (KUMAR et al., 2005)

Table 2.1 – Some examples of popular commercial heterogeneous MPSoCs from three leading smartphone companies.

	Apple A14 Bionic	**Qualcomm Snapdragon 865**	**Samsung Exynos 990**
Cores	2x Firestorm (Big cores) 4x Icestorm (Little cores)	1x Cortex A77 @ 3.1GHz 3x Cortex-A77 @ 2.4GHz 4x Cortex-A55 @ 1.8GHz	2x Mongoose 2x Cortex-A76 4x Cortex-A55
GPU	4 core (Apple in-house design)	Adreno 650	Mali-G77 11 cores
AI DSP	16-core Neural Engine	Hexagon 698 DSP + Tensor Accelerator	Dual-core NPU + DSP
Process	5nm	7nm EUV	7nm EUV

Source: (Triggs, Robert, 2021)

multi-program workload performance (KUMAR et al., 2004; KUMAR et al., 2005). The critical insight of such architectures is that application heterogeneity can be leveraged by providing the opportunity to effectively match application characteristics to the best core for a given requirement (e.g., performance or power). An example is mapping high-ILP applications to larger cores (with higher achievable IPC), and low-ILP applications to smaller cores that provide better energy efficiency.

Such architectures' effectiveness has led companies to move from the traditional homogeneous multicore settings to heterogeneous multicores. Specifically, the most notable technology companies like Samsung, Qualcomm, and Apple rely on heterogeneous MPSoCs to better accommodate application performance while keeping acceptable power consumption. For example, Samsung and Qualcomm rely on the ARM's DynamIQ technology (ARM, 2021) for the Exynos and Snapdragon MPSoCs series, respectively, which represent most MPSoCs used in mobile phones. Tab. 2.1 showcases three examples of commercial chipsets from Apple, Qualcomm, and Samsung. Most modern chips are highly heterogeneous systems composed of a cluster of different core configurations, Graphics Processing Unit (GPU), Digital Signal Processor (DSP), and a dedicated AI engine for emerging ML applications (most commonly for neural networks).

The ARM's big.LITTLE architecture was the first implemented in a commercial mobile MPSoC, which later evolved to the more flexible and efficient DynamIQ technology (ARM, 2021). Among others, DynamIQ incorporates key architectural innovations such as:

- Flexible configurations of up to eight cores. Standard configurations include up

to three different cores (big, medium, and small cores) to meet both performance, energy, and thermal efficiency goals of diverging applications;

- All cores reside in a single cluster with a coherent and shared memory. This facilitates task migration between cores through the shared memory and reduces memory traffic of shared data between different cores, increasing the performance and energy efficiency;
- Independent voltage and frequency domains for each core to tune power/performance, allowing for fine-grained DVFS to scale performance and power up or down, according to the tasks' needs;
- Enhanced power features that reduce the latency to transition between power states (i.e., ON, OFF, and SLEEP states supported by the Cortex-A series);
- An Energy Aware Scheduler (EAS) that provides fast and efficient task migration to facilitate the software-to-core mapping, providing more intelligent power and performance management.

Despite the performance and energy efficiency improvements brought by heterogeneous systems, innovations in such architectural paradigms come with design challenges due to the extra chip complexity introduced by both general-purpose cores and tailored architectures for domain-specific applications. Some of the main challenges associated with the emergence of heterogeneous designs are highlighted in the next section.

2.5 The Chipyard Framework

Continuous hardware improvement requires advanced and agile tools to design, compile, simulate, verify, and validate new architectures. Specially, the high degree of current hardware specialization and integration makes it difficult to keep up with the large number of different architectural innovations because such projects demand significant effort, time, and cost. To contextualize this claim, consider that past hardware design approaches focused on simpler general-purpose chips for generic applications, which in turn allowed for the Non-Recurring Engineering (NRE) costs (i.e., the one-time research and engineering cost) to be amortized by selling a vast number of the same chip. Nowadays, however, application-specific demands require more hardware specialization and heterogeneity that leads to increased NRE costs per chip due to the higher diversity of chip designs, making it more challenging to amortize the costs.

Therefore, hardware designs to sustain the performance demands of varying applications require constant microarchitectural innovations supported by advanced Very Large-Scale Integration (VLSI) toolchains to generate and validate new architectures. Specially, agile hardware design has been gaining more attention over the last years to alleviate the design efforts required for newer architectures.

In order to alleviate the complex VLSI design efforts and costs, the work of (Amid et al., 2020) proposes the *Chipyard* framework, which is a *generator-based agile design process for hardware* that facilitates the VLSI flow from the high-level hardware description to the final low-level circuit synthesis. Chipyard is a generator-based infrastructure that provides an integrated SoC with *reusable hardware design* (i.e., decreasing the NRE costs). It provides open source, parameterizable and modular Register-Transfer Level (RTL) designs of the Rocket Chip SoC generator (ASANOVIć et al., 2016). In essence, Rocket Chip is a collection of modular hardware designs described in the Chisel *hardware construction language*, which is a high-level and modular Domain-Specific Language (DSL) for productive/software-like hardware implementation (Bachrach et al., 2012). Among others, the Rocket Chip SoC generator provides reusable libraries that integrate cores, caches (including coherence managers), peripherals, and accelerators that can be tuned/parameterized and composed together to explore customized computing systems. As of the year 2023, Chipyard provides the following designs:

- the Berkeley Out-of-Order Machine (BOOM) core: An out-of-order (dynamically dispatched) superscalar processor;
- the Rocket core: A 5-stage, single-issue in-order core (ASANOVIć et al., 2016);
- Cache memories: Split instruction and data first-level caches and a second-level unified cache.
- Hardware accelerators: a vector architecture (Lee et al., 2014), a SHA3 accelerator and the Gemmini systolic array (GENC et al., 2019).

2.5.1 The BOOM and Rocket Cores

This work experiments with the Rocket and BOOM cores provided by the Chipyard framework. Rocket and BOOM are open-source parameterizable cores developed in the Chisel Hardware Construction Language (or HCL). Both cores implement the RV64GC variation of the RISC-V ISA. The Rocket core is a simple 5-stage in-order

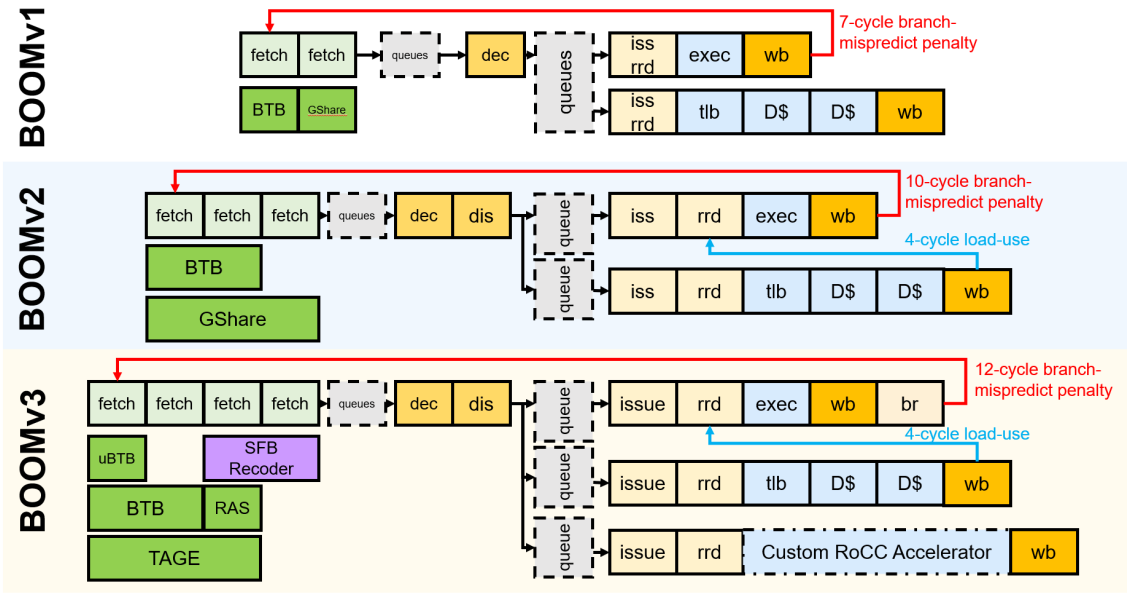
processor implementing the RV64GC variation of the RISC-V ISA. The Berkeley Out-of-Order Machine (BOOM) is a superscalar processor relying on several microarchitectural techniques to improve performance, such as out-of-order and speculative execution with advanced branch prediction.

BOOM is continuously a work in progress and has been through several microarchitectural improvements during the development of this thesis. Across its development, three branches of this processor exist: BOOMv1 (CELIO; PATTERSON; ASANOVIć, 2015), BOOMv2 (CELIO et al., 2017) and BOOMv3 (ZHAO et al., 2020). The evolution of BOOM across all its branches is shown in Fig. 2.5.

Although BOOMv1 was functional at RTL simulation, it was not physically implementable due to critical path issues as it had few pipeline stages. This issue led to the transition from BOOMv1 to BOOMv2. BOOMv2 has split and increased number of pipeline stages as well as separate instruction queues for integer, floating point, and memory access instructions. Such improvements fixed the critical path issues and made BOOMv2 physically synthesizable (the reference implementation can be found in (CELIO et al., 2019)).

BOOMv3 improved upon BOOMv2 by introducing support for compressed RISC-V instructions (RVC) and many improvements in the branch prediction and instruction fetch capabilities. For instance, the simpler pattern-based global history predictor of BOOMv2 was replaced by the more complex tagged geometric (TAGE) predictor, sup-

Figure 2.5 – Evolution of the BOOM processor organization.



Source: (ZHAO et al., 2020)

porting parallel prediction across multiple history lengths. BOOMv3 also introduced support for the Rocket Custom Coprocessor (RoCC) interface, an extension to provide support for core communication with custom hardware accelerators (some examples are mentioned in Sec. 2.5).

3 RELATED WORK

The following sections overview related works proposing optimization strategies for energy efficiency, reliability, and process variation. This chapter is divided into three sections discussing related works that are orthogonal yet complementary to this thesis. Section 3.1 discusses approaches aiming at increasing reliability. Section 3.2 discusses approaches to mitigate process variation in the context of both conventional and NTV designs. Lastly, section 3.3 contextualizes our work with respect to previous proposals.

3.1 Related Works on Addressing Soft Errors Reliability

This section highlights previous approaches to radiation-induced fault tolerance mechanisms. The section is divided into replication-based and mapping-based methodologies.

3.1.1 Redundancy-based Fault Tolerance Approaches

Microprocessor fault tolerance techniques can be roughly classified into three categories: (1) software-implemented methods, (2) hardware-implemented techniques, and (3) hybrid, which combine both aspects. Software-implemented techniques, such as (OH; SHIRVANI; MCCLUSKEY, 2002; CARDOSO et al., 2019), rely on compile-time instruction replication in the program code. The advantage of resilience-aware software implementations is their flexibility by not requiring hardware modifications.

Hybrid techniques combine aspects from software- and hardware-implemented techniques. Usually, data-flow effects are mitigated with the former, while the latter provides more robust control flow fault tolerance. Examples of such approaches are (AZAMBUJA et al., 2013; LINDOSO et al., 2017; MARTíNEZ-ÁLVAREZ et al., 2016).

Hardware-implemented techniques may adopt error correction codes (ECC), parity checking (CHENG et al., 2016b), or module replication, most often dual (DMR) or triple modular redundancy (TMR) at different granularities (SARTOR et al., 2017; VADLAMANI et al., 2010; Salehi et al., 2015; Kriebel et al., 2014). Such techniques can be tailored and selectively applied to specific processors at the cost of extra area and, in some cases, performance, leading to increased energy consumption (TONETTO et al.,

2019). DMR has also been adopted in a dynamic fashion (VADLAMANI et al., 2010), where the DMRs of particular structures are enabled/disabled at runtime depending on the current vulnerability of the structure, which is influenced by application behavior that varies across different phases of execution. Lastly, the ReStore architecture (Wang; Patel, 2005) adopts a symptom-based error detection strategy, where exceptions, mispeculated instructions, and cache misses are used as hints for fault detection. Error correction is achieved with rollback recovery to a previous checkpoint.

Software-implemented approaches based on instruction replication have been proposed specifically for superscalar processors (AUSTIN, 1999; SMOLENS et al., 2004; VADLAMANI et al., 2010; Wang; Patel, 2005; CHENG et al., 2016a). The Dynamic Implementation Verification Architecture (DIVA) (AUSTIN, 1999) implementation introduces a checker after the out-of-order core to ensure proper computation. The SHared REsource Checker (SHREC) (SMOLENS et al., 2004) reduces the hardware overhead by sharing functional units for computation and instruction checking.

Our approach towards fault tolerance is orthogonal, yet complementary, to the ones cited here. While most works on instruction and hardware replication approaches are more suitable for safety-critical applications, we do not explore such high-overhead design approaches. Instead, we propose a low-overhead strategy to improve the reliability of non-safety-critical applications by exploring hardware resources that are already present in most heterogeneous chips, i.e., provided that heterogeneous cores are available, we take advantage of task-to-core affinities in favor of improved fault tolerance.

3.1.2 Mapping-based Fault Tolerance Approaches

Application mapping is a well-known problem in the many/multicore processors optimization literature. Several approaches have been proposed to address this problem subject to different constraints (e.g., to optimize latency, performance, energy, or fault tolerance).

Considering the works that focus on reliability, the authors in (DUQUE; DIAZ; YANG, 2015) propose a fault-tolerant approach to application task scheduling/mapping that considers a reliability model capable of capturing the runtime system fault behavior and their correlation in time, allocating critical and vulnerable tasks to reliable cores. Aiming to minimize the number of faults occurring in the system and maximize application performance, the work of (ROZO et al., 2018) proposes an adaptive reliability-aware

task scheduling that considers static and dynamic analysis. While the former uses a genetic algorithm, the dynamic approach considers possible faults and their correlation with the changes in task mapping. Another genetic algorithm was proposed by (DAS et al., 2014) in the multi-objective perspective. The proposed algorithm aims to mitigate the core's aging and minimize the soft-error susceptibility. In that work, task mapping and DVFS are exploited to improve the system reliability, meeting specific energy budget constraints.

The work of (NAITHANI et al., 2017) proposes a dynamic mapping/scheduling algorithm to improve the system reliability through a proposed System Soft Error Rate (SSER) fault tolerance metric. The work considers heterogeneous systems composed of small and large cores. The proposed fault tolerance approach adopts specific hardware counters able to monitor the application reliability features, and proper application mappings are performed accordingly, at runtime, to the most suitable cores aiming to maximize the system reliability.

The authors in (Kriebel et al., 2014) assess the soft error reliability in single-ISA homogeneous processors in the Dark Silicon context by proposing an Adaptive Soft Error Resilience (ASER) approach. ASER works in both design time and runtime. At design time, ASER explores heterogeneous core designs with respect to reliability mechanisms (but homogeneous with respect to performance), such as adopting hardware redundancy in the register files, caches, or pipeline structures. The goal is to maximize the system's reliability while not violating Thermal Design Power (TDP) constraints. At runtime, considering different application vulnerabilities and the disposal of cores with custom reliability mechanisms from hardware customization, ASER applies a TDP-constrained allocation of applications to the appropriate cores aiming to address soft error resilience.

The work of (BISWAS; MUHURI; ROY, 2023) proposes energy-oriented mapping solutions on heterogeneous systems while also meeting reliability goals in the context of DVFS. However, fault tolerance is explored in the context of timing-induced failures raised by voltage scaling, neglecting the application-dependent AVF parameters in the proposed reliability model.

Our approach has the following contributions compared to the above-presented works:

- Transparency: Our approach is transparent, thus not requiring source code refactoring, code annotations, and recompilation. The work of (NAITHANI et al., 2017) is transparent but relies on greedy mapping solutions based on "trial and error" core-

to-core task migration. Given the associated overheads, this would be unfeasible for heterogeneous systems comprised of three or more core types;

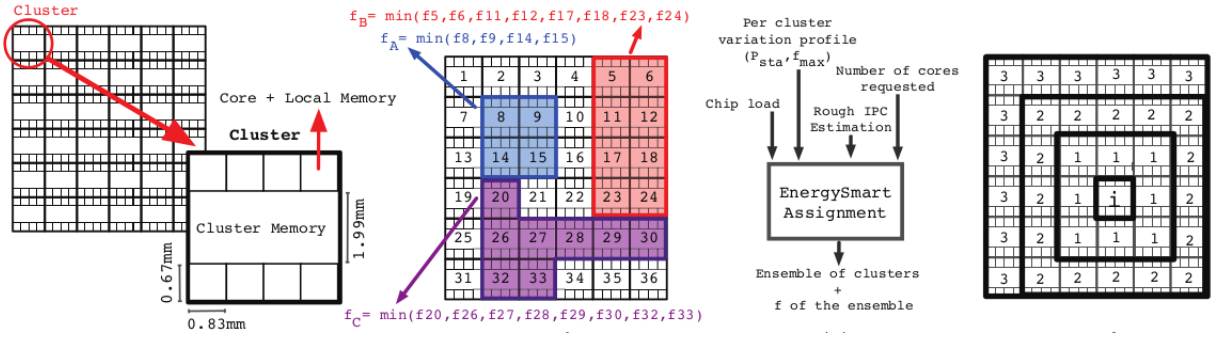
- Adaptability: Different from (DUQUE; DIAZ; YANG, 2015), which proposed application-specific optimization strategies, our work adopts a generic and run-time reliability-oriented solution. Given a set of applications, our proposal uses a trained Artificial Neural Network (ANN) to accurately estimate the application-to-core reliability-oriented *affinities* at runtime;
- Heterogeneity: We expand the design space exploration when compared to related works. For instance, the works of (DUQUE; DIAZ; YANG, 2015), (ROZO et al., 2018), (DAS et al., 2014) and (RAHMANI et al., 2017) consider only homogeneous cores. The work of (Kriebel et al., 2014) targets heterogeneous resources with respect to reliability based on hardware replication, but only performance-homogeneous cores are considered. We take a different approach by leveraging microarchitectural heterogeneity already present in most modern multicore platforms and propose a solution for reliability improvement solely based on appropriate exploration of task-to-core affinities.

3.2 Related Works on Process Variation Mitigation

Many previous works have proposed different methodologies to address process variations in the context of NTV or conventional voltage designs. In both cases, the proposals span across circuit-level mitigation strategies, e.g., static voltage level optimization (STAMELAKOS et al., 2014), fine-grained DVFS (Karpuzcu et al., 2013; Dighe et al., 2011), body biasing (ROSSI et al., 2017; GAMMIE et al., 2008), micro-architecture-level core customization (Gopireddy et al., 2016; WU; CHEN; LIU, 2023), scheduler-level approaches such as variation-aware thread mapping strategies (RATHORE et al., 2019) in conjunction with DVFS (MAITI; KAPADIA; PASRICHA, 2015; RAGHUNATHAN et al., 2013; Teodorescu; Torrellas, 2008), or cross-layer approaches (GOLANBARI et al., 2016). We provide a summary of some of such works in the next paragraphs.

In (Karpuzcu et al., 2013), the authors propose *EnergySmart*, a task assignment methodology for NTV multicore systems composed of a single voltage domain to improve area and power efficiency. EnergySmart is shown in Fig. 3.1. In this design, the chip is divided into multiple clusters, and a single worst-case frequency is assigned per cluster

Figure 3.1 – The EnergySmart NTV optimization methodology.



Source: (Karpuzcu et al., 2013)

to handle variation while also providing per-cluster/multiple frequency settings (i.e., this strategy reduces the frequency slack among cores in a cluster). The objective then consists of properly devising DVFS allocation schemes globally to effectively explore NTV in the whole chip while providing clusters with different frequency domains in which proper core assignment is adopted to increase energy efficiency. Chips operating fully at NTV, however, require a very high number of cores to sustain acceptable performance.

The work of (GOLANBARI et al., 2016) proposes a variation-aware and cross-layer optimization approach targeting Arithmetic and Logic Units (ALUs) at NTV. The key insight is that ALUs are designed to perform different types of instructions with wide delay diversity across instruction types. The approach then relies on ALU re-design, with increased clock frequency, to reduce ALU idle time of fast instructions while providing multi-cycle support for slow instructions. Fast instructions execute in a single cycle, with reduced delay, thus reducing ALU leakage energy. Process variations are mitigated by designating worst-case clock periods based on static timing analysis. The approach strongly depends on ALU re-design and on compiler support for proper code generation (e.g., by replacing slow instructions with fast ones) to better match instruction types to the proposed ALU timing constraints.

In (STAMELAKOS et al., 2014), it is proposed a variation-aware voltage island formation for NTV many cores, in which smart fine-grained voltage island configurations (layouts) are formed aiming at the minimization of the impact of within-die variation. In the work, the variation maps for a manycore are extracted from the VARIUS-NTV tool (a framework to model process variation) by considering a frequency able to keep certain performance restrictions at NTV. Then, the tool generates multiple voltage island shapes, for the given frequency, with independent V_{dd} regulators to keep up with the variation-related issues and reduce the power impact of multiple voltage domains while maintaining performance constraints.

In (RAGHUNATHAN et al., 2013), core-to-core variation and dark silicon issues in micro-architecturally homogeneous chips are considered at conventional voltage settings. An algorithm is proposed for optimal core selection, thread mapping, and frequency assignment for multi-threaded applications to reduce core-to-core variation's power/performance impact. Under a power budget, the objective is to maximize performance by exploiting process variation and cherry-picking the best subset of cores to map applications while keeping the remaining idle cores dark (power gated).

In (Dighe et al., 2011), within-die frequency and leakage variation is measured for a homogeneous 80-core processor at conventional voltage settings. A global energy optimizer is proposed for the system. A runtime energy model is proposed by considering both chip characterization (with variation profiles) and application characteristics. By monitoring application behavior (e.g., communications and compute activity), the optimum operating point for a workload is achieved by determining the number of active cores, their locations on the die, and individual voltage/frequency values, which results in better energy efficiency. A thread migration strategy is proposed in which threads with long execution cycles can be migrated to faster cores to improve performance or energy efficiency.

The work of (RATHORE et al., 2019) proposes Life Guard, a reinforcement learning-based approach for aging-aware task mapping for variation-afflicted homogeneous manycore systems at conventional voltage settings. Power and temperature are constraints for efficient online and adaptive mapping decisions aiming to maximize the system's reliability. Life Guard is dynamic and can adapt to different applications online.

The work of (Teodorescu; Torrellas, 2008) proposes variation-aware DVFS and scheduling algorithms for power management for conventional homogeneous chips, in which static power and frequency variations are considered. In order to maximize throughput (under a given power budget), variation-aware application scheduling is combined with a linear programming approach to find the best voltage and frequency pair levels for each of the cores in the MPSoC. Also, combined with variation-aware DVFS algorithms, the work evaluates several application mapping strategies that consider application behavior (e.g., IPC and dynamic power) for fine-tuning mapping algorithms to maximize performance under a power budget.

While most of the referred works only explore micro-architecturally homogeneous systems, edge architectures (e.g., mobile phones) are often highly heterogeneous chips (e.g., with at least two diverging core micro-architectures). Still, variation-related re-

Table 3.1 – Previews works on addressing varying optimization goals.

Manuscript reference	Performance	Energy efficiency	SEU reliability	PV mitigation	NTV designs	Hetero archs
Martínez-Álvarez et al. (2016)			✓			
Oh, Shirvani and McCluskey (2002)			✓			
Biswas, Muhuri and Roy (2023)			✓			✓
Azambuja et al. (2013)			✓			
Lindoso et al. (2017)			✓			
Cheng et al. (2016b)			✓			
Sartor et al. (2017)			✓			
Smolens et al. (2004)			✓			
Vadlamani et al. (2010)			✓			
Wang and Patel (2005)			✓			
Duque, Diaz and Yang (2015)			✓			
Rozo et al. (2018)			✓			
Das et al. (2014)			✓			
Naithani et al. (2017)			✓			
Kriebel et al. (2014)			✓			✓
Rehman et al. (2014)			✓	✓		
Rahmani et al. (2017)			✓			
Khdr et al. (2017)		✓				✓
Shafique et al. (2015)		✓		✓		
Stamelakos et al. (2014)		✓		✓	✓	
Karpuzcu et al. (2013)		✓		✓	✓	
Rossi et al. (2017)		✓		✓	✓	
Maiti, Kapadia and Pasricha (2015)	✓	✓		✓	✓	
Gammie et al. (2008)				✓	✓	
Gopireddy et al. (2016)				✓	✓	
Dighe et al. (2011)		✓		✓		
Rathore et al. (2019)	✓			✓		
Raghunathan et al. (2013)	✓	✓		✓		
Teodorescu and Torrellas (2008)	✓	✓		✓		
Tarsa et al. (2019)		✓				✓
Salehi et al. (2015)		✓		✓		
Golanbari et al. (2016)		✓		✓	✓	
Haghbayan et al. (2014)		✓			✓	
Stamelakos et al. (2019)		✓		✓		
Wang et al. (2017)	✓	✓		✓	✓	✓
Wu, Chen and Liu (2023)		✓		✓	✓	
This work	✓	✓	✓	✓	✓	✓

search for both NTV and STV settings remains a vastly unexplored topic in the scope of heterogeneous architectures. Therefore, we complement previous works by considering both micro-architectural heterogeneity among cores and circuit-level process variation. We mitigate variations by proposing a composite strategy (i.e., at design- and post-design time) to improve the performance and energy efficiency of heterogeneous systems while also maintaining minimum yield requirements by properly harnessing process variation of both NTV and STV cores.

3.3 Contextualizing this Thesis with Respect to Previous Works

We highlight the general optimization goals of related works in Tab. 3.1. Very few works have explored heterogeneous chips in the context of process variations, NTV, or SEU reliability. Additionally, not all works optimizing performance and energy efficiency

address process variations. Most importantly, despite the high number of publications on variation-aware chip optimization, we observe that no previous work has proposed to optimize both NTV and conventional cores in conjunction in the same die while also providing performance-optimized architectures that are gauged towards design-specific power limits while keeping yield under control. We propose that such a design strategy helps to (1) relieve the NTV frequency degradation that can otherwise only be attenuated by adopting a very high number of cores (e.g., 128 and 256 cores in (STAMELAKOS et al., 2014) and (Karpuzcu et al., 2013), respectively), and (2) provide increased performance even when compared to conventional designs due to better exploration of power slacks under the power limit.

As a final remark, we do not, however, address both SEU reliability and PV mitigation holistically, i.e., we only explore SEU fault tolerance approaches in the context of conventional designs. For NTV environments, on the other hand, reliability is addressed in the context of process variations.

4 APPLICATION MAPPING APPROACHES TO IMPROVE RELIABILITY AND PERFORMANCE OF HETEROGENEOUS SYSTEMS

In this chapter, we propose an approach to leverage application behavioral diversity in conjunction with microarchitecture-level heterogeneity aiming at improving chip-level reliability requirements. For that, we elaborate an effective learning-based method for runtime application mappings with the aim of maximizing the system's Mean Workload To Failure (MWTF). These contributions were published in (TONETTO et al., 2020).

4.1 Improving MWTF with Application Mapping

The current technology shrinking process and low-voltage operation of contemporary microprocessors pose new reliability challenges that arise from ionizing particles, such as Single-Event Upsets (or SEUs), that cause malfunction in mission-critical and non-critical applications (MITRA et al., 2014; HENKEL et al., 2013; SEEPERS; STRYDIS; GAYDADJIEV, 2012). While past reliability research has focused on single-core processors (WANG et al., 2004), the limitations in ILP have led to the emergence of heterogeneous multicore systems, such as the big.LITTLE and DynamIQ (ARM, 2021) architectures to meet the demands for more workload throughput, raising the importance of reliability improvement strategies for heterogeneous multicores.

Dealing with the adverse effects radiation-induced upsets for mission-critical applications frequently involves developing resource-consuming fault tolerance mechanisms with hardware or software redundancy - such as Error Correcting Codes (ECC), or Redundant Multi-Threading (RMT) (OZ; ARSLAN, 2019) - or hardware replication methods - such as Dual/Triple Modular Redundancy (DMR/TMR) (VADLAMANI et al., 2010; Salehi et al., 2015), or even entire processor replication such as in lockstep parallel execution manner (RODRIGUES et al., 2019). However, ECC is mainly adopted to protect storage-based structures (e.g., register files, caches, and the main memory) and does not apply to microprocessors' combinational logic and internal pipeline structures. Most importantly, the hardware replication approaches impose severe area, power, performance (e.g., DMR with rollback recovery), and energy overheads.

While in many mission-critical application scenarios such full protection methods are necessary, such approaches become less attractive for non-critical applications that

require performance and energy efficiency at restricted power envelopes. Like so, we do *not* address reliability issues concerning safety-critical applications. We take a different approach applicable to non-critical applications and do not propose resource-consuming hardware replications. Instead, we postulate that multicore heterogeneous systems provide inherent opportunities for resilience improvement without incurring severe replication overheads. Namely, such systems provide opportunities to exploit the diversity in application properties in favor of improved reliability with low cost.

Additionally, workload reliability is an issue that concerns both cloud-based and edge-based applications. For instance, server-based infrastructures are prone to large-scale Silent Data Corruptions (SDC) that are hard (and expensive) to trace and correct (DIXIT et al., 2021; DIXIT et al., 2022). At the same time, edge computing devices require low-overhead fault tolerance approaches due to stringent power and latency constraints (SEEPERS; STRYDIS; GAYDADJIEV, 2012; HOA et al., 2023; WAN et al., 2023). In this way, maximizing the amount of correctly executed workloads per number of failures encountered is attractive, especially because low-cost solutions are attainable, for instance, by taking advantage of chip heterogeneity already present in many edge- and cloud-based infrastructures.

In this part of the thesis, then, we investigate a low-overhead strategy to improve a system's MWTF, i.e., the mean computation workload that can be handled before the system experiences its next failure. In a heterogeneous system running multiple applications concurrently, the main factors that affect the MWTF are: (1) the performance of the system; (2) the Architectural Vulnerability Factor (AVF) of the cores; and (3) the raw Soft Error Rate (SER) of each core. Considering that (1) and (2) are affected by the application's characteristics and workload mapping and (3) depends on the core's area, we propose a machine-learning-based runtime methodology that factors these three aspects altogether to obtain near-to-optimal application mappings in a *transparent* fashion, aiming at increasing the system's MWTF.

Previous works such as (Kriebel et al., 2014; DUQUE; DIAZ; YANG, 2015) rely on design-time customization and static instruction-level reliability analysis for individual applications. Other runtime strategies (e.g., (NAITHANI et al., 2017)) rely on sample-based schedulers that require simulating every application in each core to estimate better mappings. We take a different approach and propose a generic and runtime reliability-oriented solution for heterogeneous multicores based on a fast ANN, which is capable of estimating accurate *affinities* of application-to-core mappings and alleviates the need

to execute the application in each core type to predict better mappings. The ANN transparently estimates realistic Register-Transfer Level (RTL) vulnerability factors based on the runtime *occupancy* of some significant pipeline structures of a core. We then exploit the benefits of heterogeneity in a case study with multiple configurations of the RISC-V Berkeley Out-of-Order Machine (BOOM) superscalar processor (CELIO; PATTERSON; ASANOVIć, 2015), considering multicore scenarios composed of *Small* (single-issue), *Medium* (dual-issue), and *Large* (quad-issue) dynamically scheduled cores. The key contributions of this part of the thesis are as follows:

• We propose a runtime and transparent method that finds close-to-optimal application mappings in heterogeneous systems such that the resilience (and increase in workloads per joule before the next failure) is maximized. It is adaptive and supports any new incoming application without needing compile-time information about the application characteristics;

• We experiment with heterogeneous multicores in a realistic scenario by performing 2.4 million fault injections, gathering accurate (RTL) vulnerability factors, and comparing the achieved resilience improvement against traditional homogeneous architectures.

4.1.1 Motivation and Background

The primary motivation behind the proposed work is based on the observation that application reliability is affected not only across different heterogeneous configurations but also across different application-to-core mappings. To support this claim, we explore heterogeneous systems composed of different numbers of *Small* (single-issue), *Medium* (dual-issue), and *Large* (quad-issue) cores, shown in Table 4.1. The *Medium* and *Large* cores resemble the ARM Cortex-A9 and Cortex-A15, respectively. In contrast, the *Small*

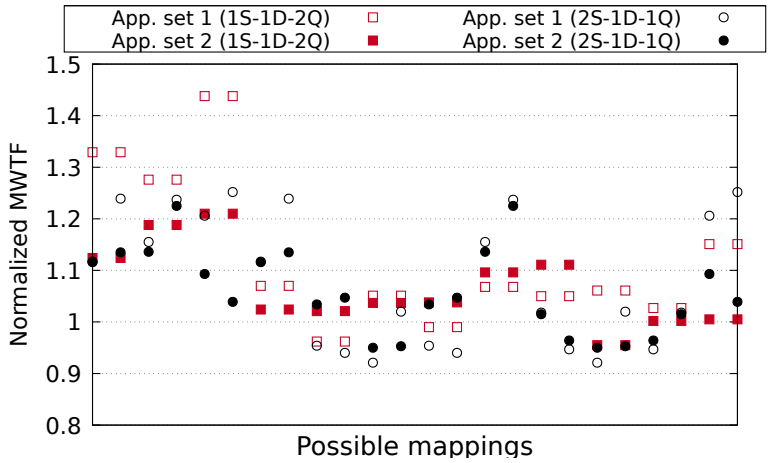
Table 4.1 – Explored core configurations.

	Small (S)	**Medium (D)**	**Large (Q)**
Fetch-/Issue-/Commit-width	1	2	4
Physical Register File	100	110	128
ROB entries	24	48	64
Issue Unit entries	10	20	28
Load/Store queue entries	4	16	32
Area (mm^2)	0.08	0.12	0.20

core is adopted to increase design space exploration and heterogeneity as it can improve area and energy efficiency. For the sake of generality, we evaluated all three possible heterogeneous multicore configurations composed of three different cores, which are: one single-issue, one dual-issue, and two quad-issue (1S-1D-2Q); one single-issue, two dual-issue, and one quad-issue (1S-2D-1Q); and, two single-issue, one dual-issue, and one quad-issue (2S-1D-1Q). In the following sections, we also experiment with homogeneous configurations composed of only *Medium* (4D) and only *Large* (4Q) cores. In all of the obtained results, the reported data is normalized to the baseline homogeneous configuration composed of only *Small* processors (4S).

For the configurations 1S-1D-2Q and 2S-1D-1Q, Fig. 4.1 depicts the normalized MWTF, computed as detailed in section 2.2, for two sets of distinct applications. Each set contains four applications (explained in section 4.3), and the results are depicted across all 24 possible mappings (x-axis) of the four applications mapped to the four core types. From the figure, we can conclude that the MWTF in a heterogeneous system varies significantly across different application mappings (e.g., up to 49.4% for the set of applications 1 in the configuration 1S-1D-2Q) and across different configurations. We then exploit the fact that better MWTF can be achieved by using smart application mappings with low cost.

Figure 4.1 – MWTF obtained for two application sets for all possible mappings.



4.2 Adaptive Mapping

4.2.1 Problem Definition

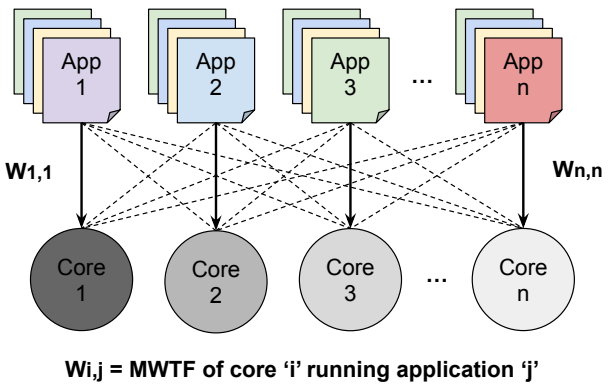
This part of the work aims to propose a learning strategy to increase the chips' MWTF. As described in section 2.2, and recalled in Eq. 4.1, the MWTF is the amount of workload computed before a failure is experienced in the system.

$$MWTF = (SER_{raw} \times AVF \times execution\,time)^{-1} \propto IPC/(SER_{raw} \times AVF) \quad (4.1)$$

To increase the MWTF in a heterogeneous system, we seek an application mapping that: (1) reduces the total *execution time* while not significantly increasing the AVF, or (2) reduces the overall system's *AVF* while not impacting too much in performance. The third variable involved in optimizing MWTF is the raw SER. However, we assume it to be constant for each core type since, as already discussed, it depends entirely on the chip area, the manufacturing process, and the environment. Considering that, we leverage the heterogeneity of the multicores to find application mappings that yield proper balance between *execution time* and *AVF* such that the system's MWTF increases.

In a heterogeneous multicore system, such a mapping task can be thought of as the *Assignment Problem* (EDMONDS, 1965), in which the application-to-core mapping is modeled as a weighted bipartite graph, as in shown Fig. 4.2. In this model, each set of vertexes represents applications and cores, respectively, and the graph's edges represent the *affinity* of mapping each application to each core. As can be seen in Fig. 4.2, each weight expresses the trade-off between Instructions Per Cycle (IPC) and the effective

Figure 4.2 – Assignment graph of application mappings aiming to maximize the overall MWTF.



failure rate of the core, where the effective failure rate is estimated by the product of the raw SER and the AVF (derating) factor.

Therefore, In the proposed model, an edge with a high weight value that maps an application to a core means the application is likely to have a high MWTF. Our mapping strategy, then, lies in estimating the values of each edge in the mapping graph and then applying a proper algorithm to find the best set of edges that increases the overall affinity of the system's application mapping by solving *maximum weighted perfect matching problem*, which is solvable in polynomial time (i.e., it is scalable for a higher number of cores) as proposed in the work of (EDMONDS, 1965).

The methodological flow of our strategy is depicted in Fig. 4.3. Our approach for fast runtime mapping relies upon executing the applications in the quad-issue (more complex) core and then inferring how the same applications would perform, in terms of AVF and IPC, in the simpler cores (single- and dual-issue types). To estimate the values of the edges for each application in the assignment graph, we propose two ANN designs to predict AVF and IPC of the given applications when mapped to a heterogeneous design. The approach is divided into a design-time phase, where we train the ANNs, and a runtime phase, where we perform ANN inference to build the assignment graph and run the mapping algorithm.

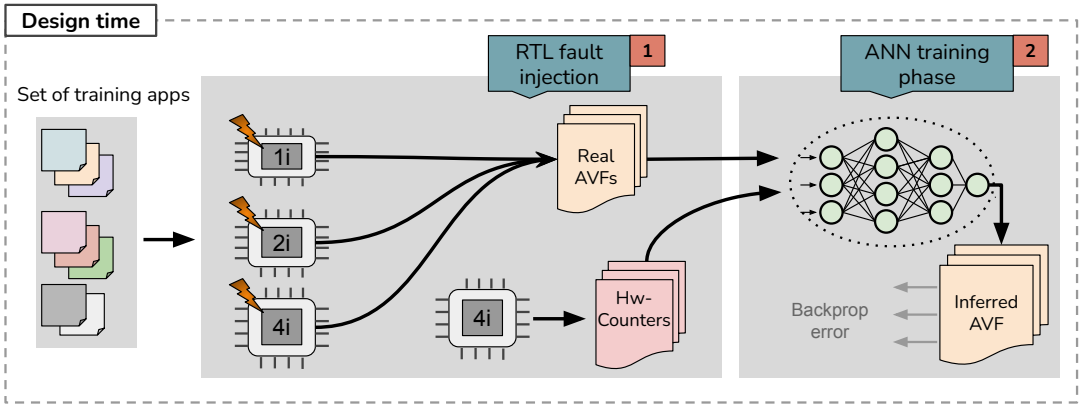
To estimate the core's AVF for a given application, we train an ANN based on previous accurate AVF metrics estimated during design time with fault injection in RTL models for each core (the fault injection process is further elaborated in section 4.3). Additionally, the IPC of all applications for all core types is measured with performance counters at the cycle-accurate level. We collect the IPCs of every application and train another ANN that receives the quad-issue's IPCs as input and outputs the IPC for the other core types, as detailed next.

During deployment, for the online estimation of the AVF, the AVF neural network receives as input the *occupancy* of the quad-issue core structures and outputs an inferred core AVF for each core type. Similarly, the IPC estimation consists in measuring the quad-issue's IPC and feeding it to the ANN, which outputs the IPC estimation for the same applications in the other core types. The process of monitoring the occupancy of the structures, estimating the AVF and IPC, and then using it to perform the application mapping is transparent, and no previous compile-time information about the application characteristics needs to be informed to the mapping strategy.

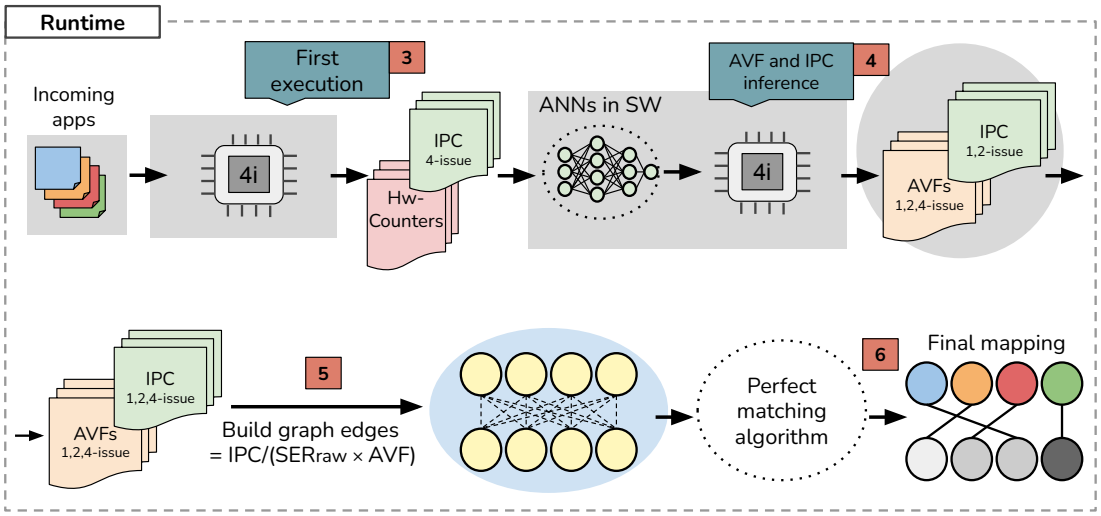
The details of the design-time and runtime approaches, described in Figs. 4.3a

Figure 4.3 – The predicted AVF estimation by the neural network compared to the expected values estimated with fault injection.

(a) Desing-time ANN training approach



(b) Runtime ANN inference and mapping



and 4.3b, respectively, are as follows:

Design time: Consists of training the ANN, as shown in Fig. 4.3a, according to the following steps:

Step 1: We first gather realistic AVF estimations for a set of *training* applications by running them on each core type and performing statistical fault injection to simulate soft errors (single bit-flips) at the RTL. We also simulate the quad-issue processor for each training application in order to track the occupancy of four significant structures in the pipeline: the Physical Register File (PRF), the Reorder Buffer (ROB), the Load/Store Unit (LSU), and the Issue Unit (IU). The choice for monitoring such structures comes from the fact that they hold most of the microprocessor state, such that their occupancy is highly correlated with the core's AVF.

Step 2: The data gathered in the previous step are used to train the ANN that re-

ceives the quad-issue's structures' occupancy as inputs. The outputs of the training neural network are the AVF estimations for the single-, dual- and quad-issue cores. For each application, we derive the AVF of the three core types based only on the occupancy of the quad-issue processor so that a single ANN is required, and we avoid having to execute each application in each core to take the final mapping decision. We empirically verified that using the occupancy of the quad-issue processor provides better accuracy in estimating the AVF for all core types. We train a second ANN that receives the application's IPC in the quad-issue core and outputs the IPC for all other core types.

Runtime: During execution time, shown in Fig. 4.3b, the core assignment to a new incoming application is transparently estimated in a process that consists in calculating the value of each possible edge (to the three core types), according to the formula in Fig. 4.2, and then applying the maximum weighted perfect matching problem to find the best possible mapping such that the sum of the selected weights is maximized. For that, it is necessary to know the AVFs and IPCs of each application. The raw SER for each core is constant and application-independent, so estimating it during runtime is unnecessary. The value of each edge is estimated as follows.

Step 3: The AVF of each application in each core is estimated by the ANN solely based on the current degree of occupancy of the quad-issue processor so that we avoid executing each application in each core to estimate their AVFs. It is then necessary to first execute each application in the quad-issue processor for a fixed number of cycles in a sample phase and log the occupancy of each application. The choice for the number of sample cycles to simulate is adjustable, and designers could extend such an approach to capture desired phases of execution better. We experimented by simulating the whole application to model a request-based scenario where the same set of applications is repetitively executed. That is to say, one workload corresponds to a complete execution of a given application. Notice that for any given application that executes repetitively, the ANN inference has to occur only once per application.

The occupancy of each structure is measured in a cycle-accurate fashion: for each structure, we adopt a special counter register that counts the number of busy entries in the structures during the sample phase. The PRF's occupation is measured by monitoring the physical register's *free-list*, which informs the number of physical registers in the PRF that are allocated to architectural registers (the BOOM core relies on register renaming to resolve output dependencies (WAW) and anti-dependences (WAR)). The other structures are monitored by probing specific *valid* and *busy* signals in each entry. Each cycle,

the counter registers are incremented (or decremented) whether a new entry in the corresponding structure is allocated (or deallocated). After the sample phase finishes, the calculation of occupancy of each structure is then performed by dividing the accumulated number of allocated entries by the total number of entries in the structure multiplied by the number of cycles executed in the sample phase, i.e., it calculates the average number of busy entries over the whole application execution.

The IPC of an application in each core type is gathered during the sample phase (i.e., the application running on the quad-issue) by monitoring the quad-issue's IPC with performance counters. Because IPC is affected by several micro-architectural factors, we then use an ANN to estimate the IPC of the same application in the single- and dual-issue cores.

Step 4: After the application ends its first execution and the occupancy for the structures is gathered, the quad-issue core executes the software-implemented ANN to estimate the AVF of the three processors. We experimentally evaluated several ANN topologies and concluded that an ANN with three hidden layers, six neurons each, yielded better AVF estimations (as shown in section 4.4.5). We then estimated that the overhead of executing such software-based ANN in the quad-issue core is approximately 8k cycles (estimated with the cycle-accurate simulator).

Steps 5 and 6: Finally, the scheduler is provided with the estimated AVFs and IPCs of each application mapped to each core and builds the bipartite graph that is used as input to the maximum weighted perfect matching algorithm to solve the assignment problem and estimate the proper application mapping which maximizes the sum of the assigned edges.

4.3 Experimental Methodology

Our experiments consist in first gathering information to train the ANN (design time) with a set of training applications and then evaluate the results (runtime) with a set of test applications. Because we want to completely separate test and training applications to avoid a biased ANN (RUSSELL; NORVIG, 2009), we adopt K-fold cross validation to train the ANN with all applications, excluding the test one. Each training set contains seven applications, leaving one application for testing.

In order to train the ANN, we first perform statistical fault injection that simulates SEU in the three configurations of the BOOM processor (Small, Medium, and Large).

Error injections can be masked during execution or cause application-level failures. Failures are classified into three categories: (1) Silent Data Corruptions (SDC), timeouts (due to control faults), or (3) simulation crashes.

We adopt the statistical model in (LEVEUGLE et al., 2009) to obtain statistically reliable AVF estimations for the three processors. Due to the time-consuming fault injection experiments at RTL, we consider eight small benchmarks taken from the MiBench suite (GUTHAUS et al., 2001) (Median filter, Rijndael, String search, K-means, QSort, Sobel, Susan, and Dijkstra). A total of 800k faults were injected in each processor (100k per application, 2.4 million in total) at RTL to obtain statistical significance in our results, leading to a confidence level of 99% with an error margin of 1%. The fault injection campaign takes approximately three weeks. We estimated the area and power for each core configuration by synthesizing BOOM with Cadence RTL Compiler with NanGate's 15nm standard cell library (MARTINS et al., 2015) and a 2.1GHz synthesis target.

The fault injections were performed in all microarchitectural structures of the processors at RTL. However, we do not perform fault injections in the caches because we assume they can be protected with ECC. Also, the caches are not shared among the cores, and we experiment with completely independent applications so that there is no interference among them.

We evaluated our mapping strategy on the three heterogeneous multicore configurations composed of four cores with three core types, as shown in Tab. 4.2. We also compare the prediction-based MWTF to the three homogeneous configurations shown in the table. We consider a total of eight applications, which allow for a total of 70 combinations of four different applications. Each combination is an application set, and each set of four applications can be permuted (or mapped) in 24 possible ways, considering a system with four cores. For simplicity, we disregard the fact that some of the permutations are

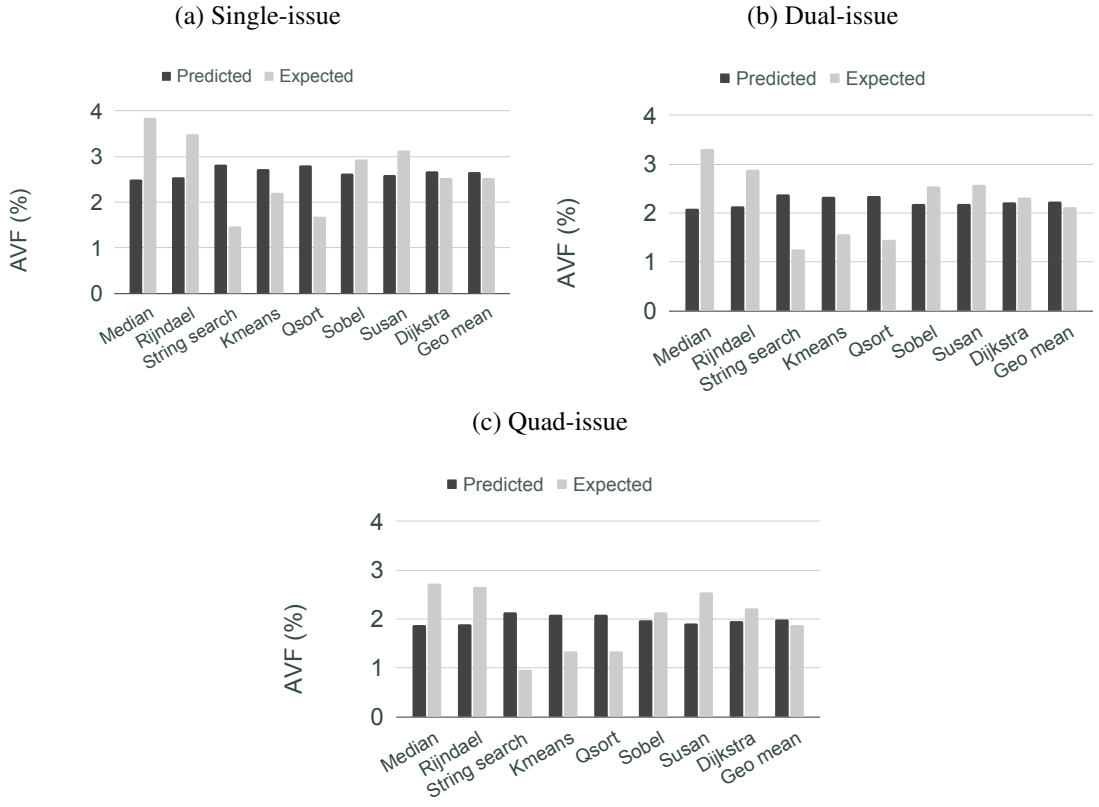
Table 4.2 – Area (mm^2) and approximate normalized flip-flop raw SER (based on the number of flip-flops) of the explored chips

	Area (cores only)	**Area (total)**	**Flip-flops (cores only)**	**Normalized raw SER (cores only)**
4S (baseline)	0.032	0.17	172652	1.00
4D	0.48	0.62	276360	1.60
4Q	0.8	0.94	385196	2.23
1S-1D-2Q	0.528	0.69	304851	1.77
1S-2D-1Q	0.448	0.59	277642	1.61
2S-1D-1Q	0.336	0.47	251715	1.46

redundant since at least two cores in the evaluated heterogeneous systems are of the same type.

For all multicore configurations, we first measured the MWTF for all different permutations for each application set and exhaustively searched for the best-case (oracle) and worst-case possible schedules that lead to the best and worst MWTF, energy, and MWTF/energy, for each application set. Then, each set of applications is again simulated by mapping the applications according to the *matching problem* that takes into consideration the ANN's predicted AVFs and IPCs (predicted-case mapping). For each application's estimations, we consider an ANN that is trained without previous knowledge on the application in order to model a scenario of unforeseen applications, i.e., we evaluate a scenario that supports transparent adaptability to new incoming scenarios.

Figure 4.4 – The predicted AVF estimation by the neural network compared to the expected values estimated with fault injection.



4.4 Results on Reliability-Oriented Mappings

4.4.1 AVF Prediction

We first show the results for the predicted AVF estimations for the single-, dual-, and quad-issue processors in Fig. 4.4. Notice that even though the geometric mean of the predicted AVFs is very similar to the expected ones, there are notable deviations in AVF estimation for some applications. However, because different applications are “competing" to be mapped to a core, our scheduling policy only requires that the proportion between the predicted and expected application AVF is kept roughly constant across different cores. For instance, consider the *Median* application. Even though its predicted AVF is considerably smaller than the expected one, such prediction is also smaller for all cores in a similar proportion, which has minor effects considering that we model the scheduling police as a *matching problem*.

Figure 4.5 – Configuration 1S-1D-2Q

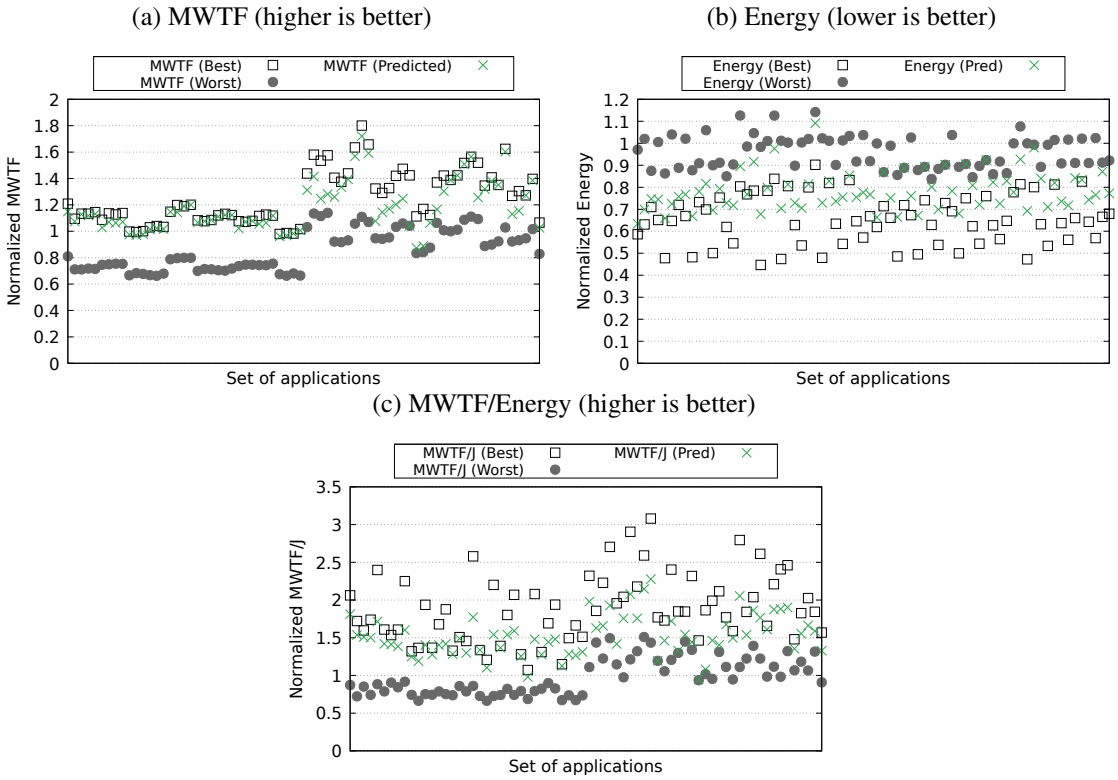
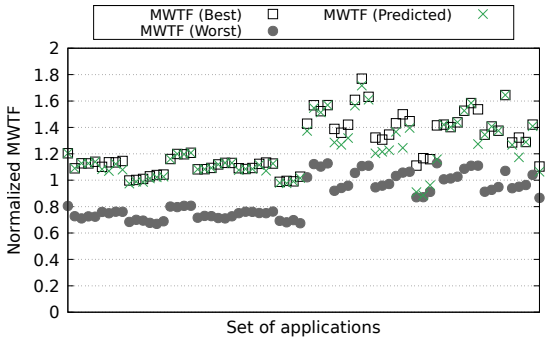
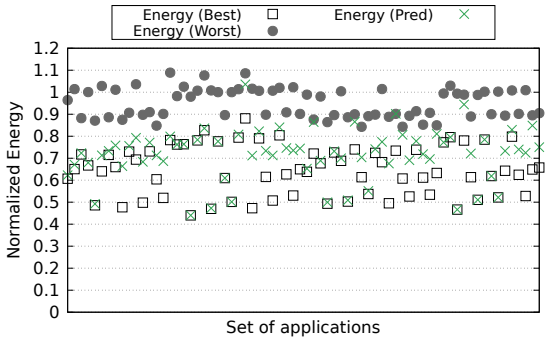


Figure 4.6 – Configuration 1S-2D-1Q

(a) MWTF (higher is better)



(b) Energy (lower is better)



(c) MWTF/Energy (higher is better)

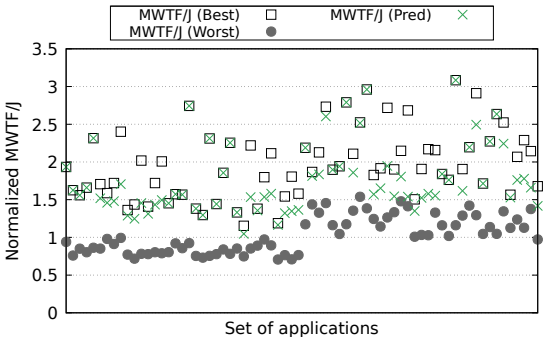
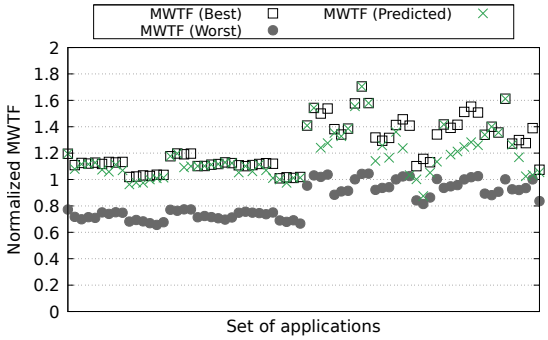
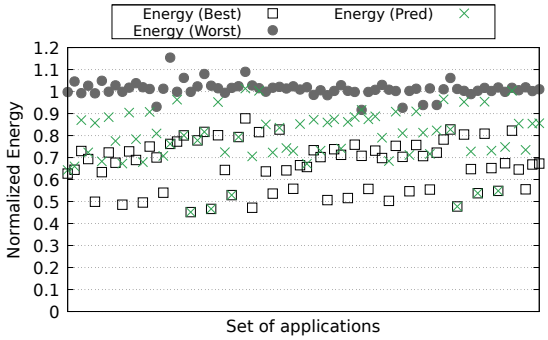


Figure 4.7 – Configuration 2S-1D-1Q

(a) MWTF (higher is better)



(b) Energy (lower is better)



(c) MWTF/Energy (higher is better)

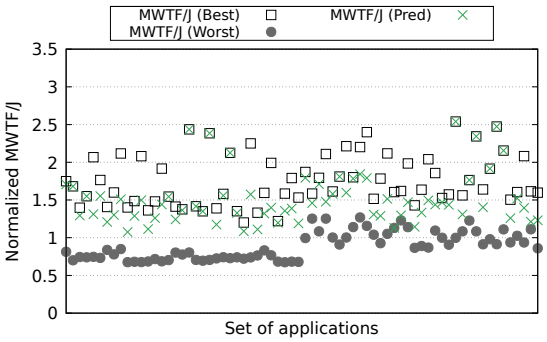
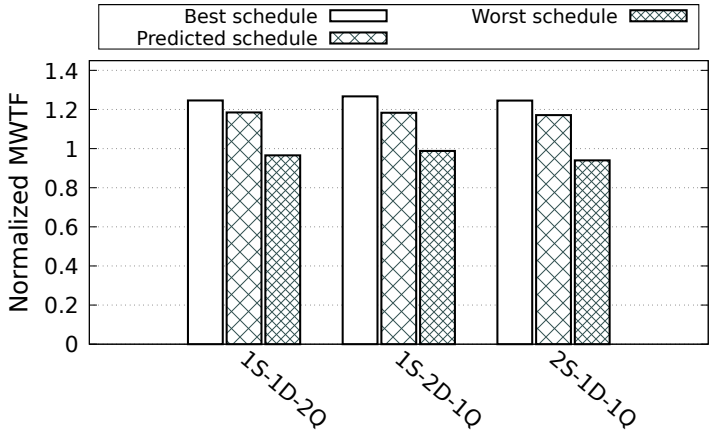


Figure 4.8 – Average MWTF gains for all application sets for the three mapping cases.



4.4.2 Dynamic Mapping Evaluation

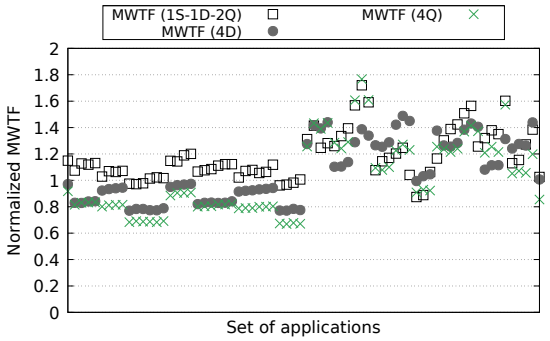
In this section, we evaluate the efficacy of our proposed dynamic mapping approach (i.e., the prediction-based mappings) applied to heterogeneous systems. For that, we evaluated the three heterogeneous architectures (i.e., 1S-1D-2Q, 1S-2D-1Q, and 2S-1D-1Q) by measuring the MWTF, energy, and the tradeoff MWTF/energy for the three mapping cases: best-case mapping (oracle), predicted-case mapping, and worst-case mapping. The metrics herein considered are normalized to the baseline configuration (only single-issue cores - 4S). Our intent is to show that the predicted-case mappings achieve chip-level MWTF very close to the oracle with small energy overheads.

The results for the three chip configurations are depicted in Figs. 4.5, 4.6, and 4.7, showing the achieved results for MWTF, energy, and MWTF per Joule (sub-figures a, b, and c, respectively). From the figures, it is clear that the predicted mappings closely match the best-case one. Because both MWTF and energy are directly correlated to execution time, we can see that the predicted schedule that aims at increasing MWTF also provides lower energy when compared to the worst-case energy schedule, as shown in Figs. 4.5b, 4.6b, and 4.7b. As a consequence, the fraction of MWTF/Joule also tends to be considerably better than the worst-case schedule, as shown in Figs.4.5c, 4.6c and 4.7c. The key observation from such results is that we can improve the MWTF without imposing significant energy overheads.

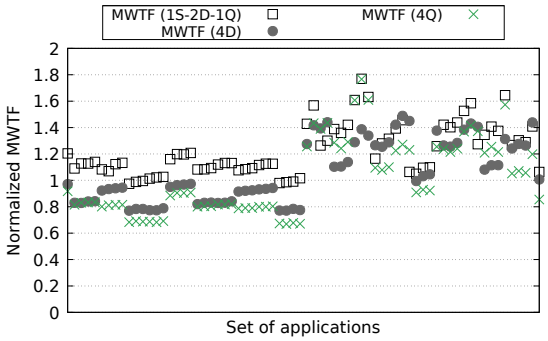
The average MWTF results for all configurations for the three mapping cases is depicted in Fig. 4.8 (i.e., the average MWTF gains for all application sets). For the configurations 1S-1D-2Q, 1S-2D-1Q, and 2S-1D-1Q, the deviation of the predicted MWTF from the oracle MWTF is around 4.9%, 6.6%, and 5.9%, respectively.

Figure 4.9 – Comparison of the three different heterogeneous configurations (using the predicted mappings) against two homogeneous ones.

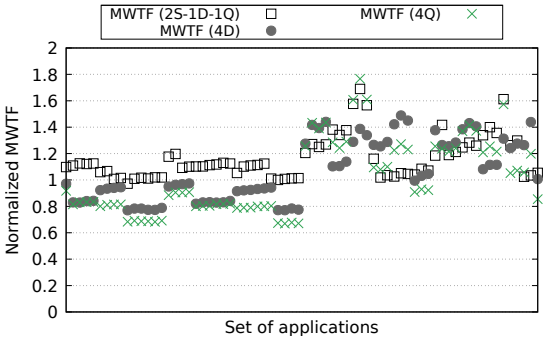
(a) Config 1S-1D-2Q vs homogeneous



(b) Config 1S-2D-1Q vs homogeneous



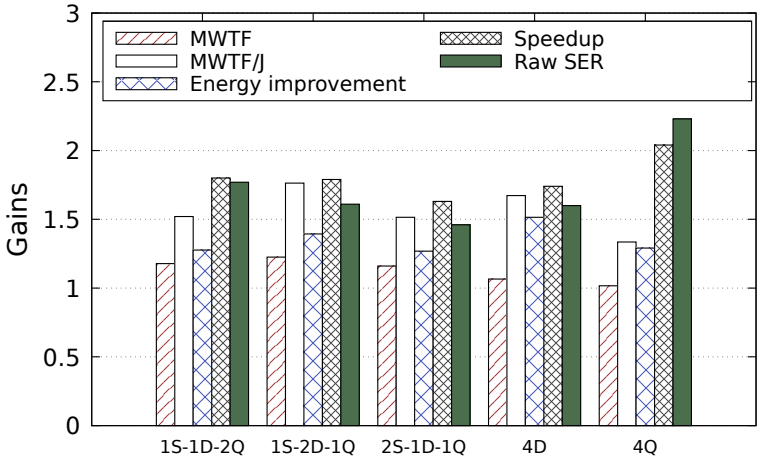
(c) Config 2S-1D-1Q vs homogeneous



4.4.3 Comparing Heterogeneous versus Homogeneous Configurations

In order to highlight how heterogeneity can improve MWTF (with prediction-based mappings), we also compared the MWTF (and also energy) of the three heterogeneous architectures to two homogeneous ones: four dual-issue cores (4D) and four quad-issue cores (4Q). For each heterogeneous configuration, we estimate the values by considering the predicted mappings for all application sets and report the average normal-

Figure 4.10 – Comparison (average values) against two homogeneous architectures (4D and 4Q).



ized improvements for each set in Figs. 4.9 and 4.10, as follows:

MWTF: While the three heterogeneous architectures provide very similar MWTF, on average, they all provide higher MWTF than the homogeneous configurations. That happens mainly due to the lack of variability within the homogeneous configurations that do not provide means for application-to-core mappings that better trade AVF for performance. For instance, even though 1S-1D-2Q has a higher number of flip-flops subject to errors when compared to 4D (a difference of 10.3% in raw SER), it better trades AVF for performance due to the quad-issue core (higher performance and lower AVF). The net result, in general, is higher MWTF for the heterogeneous configurations. The 2S-1D-1Q configuration, however, has higher MWTF than the homogeneous configurations due to its smaller area that compensates for its low performance due to the two *Small* cores. For the homogeneous configurations, even though the 4Q configuration has better performance than the 4D one, its MWTF is lower due to its larger area.

Energy and MWTF/energy: Only the configuration 1S-2D-1Q provides a better tradeoff of MWTF/energy when compared to the 4D configuration. The configuration 2S-1D-1Q has lower energy improvement due to the longer execution times caused by its two *Small* processors, which in turn reduce the MWTF/energy. On average, heterogeneous configurations provide 19.7% higher MWTF/J when compared to the 4Q (4 quad-issues) configuration.

4.4.4 Distribution of best configurations

In this section, we highlight how hardware heterogeneity yields the best possible MWTF for different application sets. For that, in Table 4.3, we show the distribution of the best possible multicore configuration for each application set (by considering the ANN-based mapping for the heterogeneous settings). The numbers show, for instance, that 62.8% of the application sets have higher MWTF with configuration 1S-2D-1Q. Also, the importance of heterogeneity can be highlighted by the fact that when the ANN-based application mapping is adopted, 87% of the application sets have higher MWTF in heterogeneous configurations.

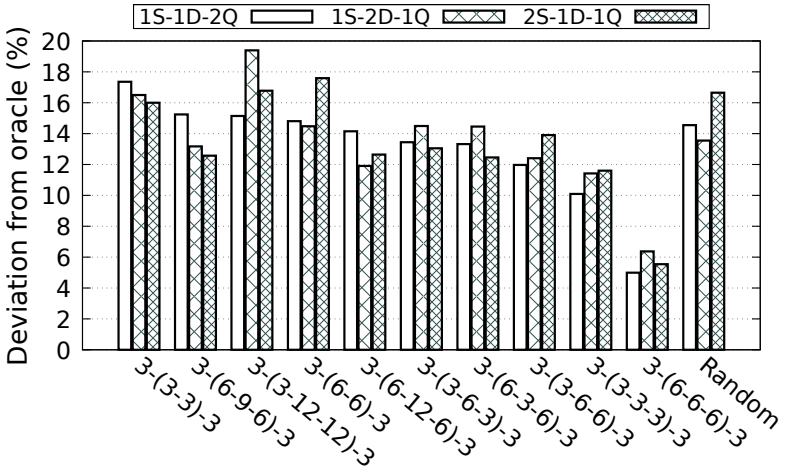
Table 4.3 – Percentage of application sets that maximize a given metric when prediction-based mapping is applied to each evaluated chip configuration. Example: 62.8% of the application sets achieve the highest MWTF with configuration 1S-2D1-Q.

	MWTF (highest)	**Energy (lowest)**	**MWTF/energy (highest)**
1S-1D-2Q	7.1%	0%	0%
1S-2D-1Q	62.8%	32.8%	48.5%
2S-1D-1Q	17.1%	10%	14.2%
4D	12.8%	55.7%	34.2%
4Q	0%	1.4%	2.8%

4.4.5 Exploiting Different ANNs

The correlation between pipeline occupancy and the overall core AVF is complex and non-linear, making selecting a proper ANN topology challenging. This process is not trivial (and often cumbersome), so the search for the proper ANN topology adopted in the previous results was performed empirically. For that, we evaluated the effectiveness of ten different ANN configurations by comparing their efficiency as measured in terms of distance of achieved MWTF w.r.t the optimal mapping strategy. We also compare the prediction-based mappings against random mappings. For the three heterogeneous configurations, Fig. 4.11 shows the MWTF deviation from the optimal mapping for the different ANNs and the result of the random mapping. Configuration 3-(6-6-6)-3 outperforms all other topologies for the three heterogeneous configurations, with an average deviation of 5.6% from the oracle while significantly outperforming the random mapping.

Figure 4.11 – MWTF deviation from oracle for different ANN configurations (lower is better).

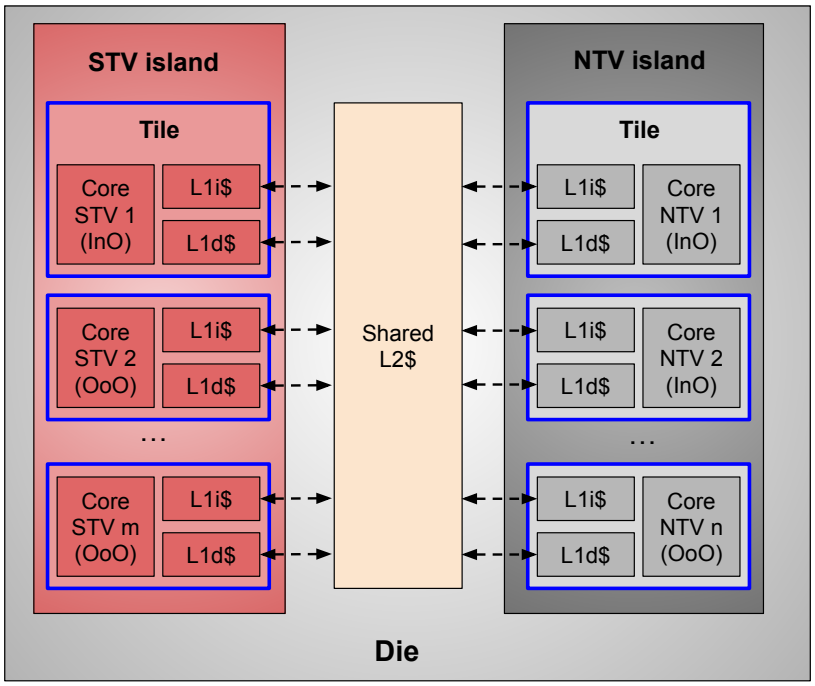


5 A POWER-EFFICIENT AND PERFORMANCE-ORIENTED EXPLORATION METHODOLOGY WITH NTV CHIPS

5.1 Introduction

Edge computing systems are becoming increasingly popular due to their ability to efficiently process data at the network edge, enabling faster response times and increased security by reducing the need for data transmission to centralized servers (Shi et al., 2016). However, these systems are often constrained by power budgets (e.g., due to battery dependence or limited cooling capabilities) that limit achievable performance. As computation is delegated to the edge side, improving microprocessor performance under strict power limits is essential for this domain. We provide an efficient NTV-based chip design approach to improve performance under such scenarios. A preliminary version of the framework presented in this chapter was published in (TONETTO; BECK; NAZAR, 2022).

Figure 5.1 – Architecture-level view of the chip.



5.2 Chip Architecture Exploration Scope

In this part of the work, we consider architectural chip exploration with both STV and NTV voltage operation points in the same chip. Fig. 5.1 shows an abstract architectural-level view of a proposed chip. Notice that the chip shown in the figure is abstract and generic. In practice, we explore several different heterogeneous architectures with diverse numbers and types (microarchitectures) of cores.

Each chip is a die comprising two voltage islands (STV in red, NTV in gray). An island is composed of multiple tiles (highlighted with blue borders), and each tile is composed of a single core plus private first-level instruction (L1i$) and data (L1d$) caches. The cores are heterogeneous, encompassing different organizations such as scalar in-order execution (InO) or different parameterizations of superscalar out-of-order execution cores (OoO). An inclusive (instructions and data) second-level cache (L2$) is shared among all tiles in the chip.

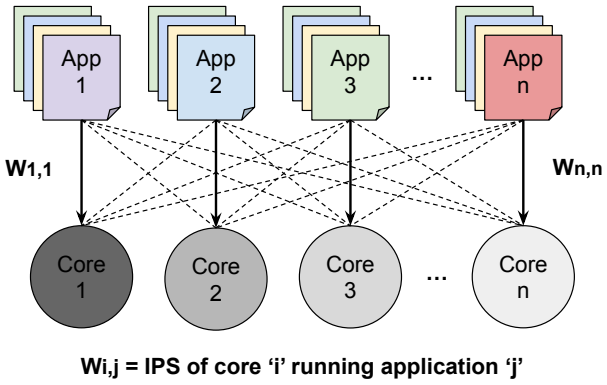
Our proposal, termed *Selective NTV Architectures for Power-efficient Edge computing* (or SNAP), consists in dividing the die into two voltage islands, one at STV, and the other at NTV, and efficiently determining what core types (and number) are the most suitable in each island. Each voltage island follows a Single-Voltage Multiple-Frequency (SVMF) approach in which all tiles in an island operate at the same voltage. Other works have adopted the chip-level SVMF approach in the context of STV (Dighe et al., 2011) or NTV (Karpuzcu et al., 2013; TARSA et al., 2019) designs. We consider per-island SVMF, with separate but unique V_{dd} settings per island (STV at 0.8V, NTV at 0.55V). However, the cores operate at individual frequency levels that are the most suitable according to the core type. The first-level caches operate at the same voltage level as their respective core. The second-level cache operates at STV level unless all tiles in the die operate at NTV.

5.3 Application Mapping with Heterogeneous Systems

In this part of the thesis, we consider chips with heterogeneous cores that vary in both Instructions per Cycle (IPC) and attainable frequency. The general optimization goal of this thesis, then, consists of finding optimal sets of cores (types and voltage islands of each core) that maximize the total chip's instruction throughput under arbitrary and user-defined power limits.

Because the chip is composed of heterogeneous cores that vary in both IPC and

Figure 5.2 – Assignment graph of application mappings aiming to maximize the overall MIPS.



frequency, a proper performance metric must be considered to encapsulate both parameters. This way, we target chip designs with optimized Instruction per Second (IPS) (because $IPS = IPC \times frequency$) while not violating the power limits. Secondly, we use IPS as a figure of merit because we want to compare achievable multi-task throughput across different chip designs that vary in the number of cores in the chips (besides IPC and frequency), i.e., we want to compare chips with diverging numbers of cores. Thus, IPS can be used as a global metric that encapsulates the effects of the proposed methodology on overall performance.

Our mapping solution models the mapping problem as a bipartite graph that maps applications to heterogeneous cores, as shown in Fig. 5.2. In this model, applications and cores are nodes that constitute a *bipartite graph*, with the edges representing the IPS of each application when mapped to each core (nodes), as shown in the figure. The goal then consists in finding the optimal mapping that maximizes the overall system's throughput (i.e., optimally maximizing the sum of the selected edges). The solution to this problem is optimized by solving the *maximum weighted perfect matching problem* on the assignment graph, optimally solvable in polynomial time (EDMONDS, 1965).

5.4 Architectural Search and Optimization Goal

The optimization goal is to find optimized chip configurations to meet the criteria shown in Eq. 5.1. For the user-defined power budget (P_b) and a set of configurations ($configs$), we seek the chip configuration ($target\_config$) with the highest IPS under the power limit.

$$TargetConfig(P_b) = \underset{conf \in configs}{\arg\max} \{IPS_{conf} | Power_{conf} \leq P_b\} \tag{5.1}$$

Our exploration space considers RISC-V heterogeneous chips with four core types, as listed in Tab. 5.1. We consider architectures ranging from 2 to 16 cores while also considering both STV and NTV versions of each core, amounting to eight distinct core types that can be chosen for the desired power limit.

For comparison purposes, our methodology considers three chip design cases, in which the chips are built with 1) all cores at STV (*Full STV* chips), 2) all cores at NTV (*Full NTV* chips), or 3) a blend of STV and NTV cores (*SNAP* chips). For the full STV/NTV cases, we consider chips formed with all possible core combinations from 2 to 16 cores, so the number of MPSoC configurations is $\sum_{i=2}^{16} \binom{i+4-1}{4} = 4840$[1], making exhaustive search feasible. The SNAP/mixed configurations composed of both STV and NTV cores, however, result in a much larger design space, with a total of $\sum_{i=2}^{16} \binom{i+8-1}{8} = 735,462$ possibilities.

We do not make exhaustive exploration across all SNAP possibilities. Rather, we introduce an Integer Linear Programming (ILP) approach to efficiently compose optimized MPSoC configurations. This approach prunes the exploration space of configurations and avoids exhaustive searches while also providing efficient configurations for strict power requirements. This is analogous to the *0-1 Knapsack Problem*. Here, knapsack items are core types. Item values and weights are the cores' IPS and power dissipation, respectively, and the knapsack capacity is the maximum allowed chip power dissipation (power budget).

$$\forall \underset{core\ types\ c}{\text{maximize}} \quad \sum_c \mu_{IPS,c} \times Count_c \tag{5.2a}$$

$$\text{subject to}$$

$$\mu_{pwr,L2} + \sum_c \mu_{pwr,c} \times Count_c \leq P_b \ , \tag{5.2b}$$

$$2 \leq \sum_c Count_c \leq 16 \tag{5.2c}$$

The goal is to maximize the IPS performance (objective) under power budget limits (constraint).

[1]The number of combinations of r items (number of cores in the chip) out of n item types (types of cores, allowing repetition of core types) is expressed as $\binom{n+r-1}{r}$

The objective and constraints are depicted in Eqs. 5.2a, 5.2b and 5.2c. The inputs are the power budget limit (P_b), the average core IPS and power ($\mu_{ips,c}$, $\mu_{pwr,c}$) of each core type c, and the second-level cache average power ($\mu_{pwr,L2}$). The ILP method aims to select the optimal set of cores (outputs are the types and number of each core type - $Count_c$) to maximize the IPS performance while not violating the given power budget (P_b), as is illustrated in Eq.5.2a (budget constraint in Eq. 5.2b). We also constrain the number of cores to the range [2,16] (constraint in Eq. 5.2c).

5.5 Results on Performance-Oriented Mappings

5.5.1 Experimental Methodology

We conducted our experiments with the core implementations provided by the Chipyard open-source framework (Amid et al., 2020). The adopted core configurations are the RISC-V Rocket (in-order) core as well as three configurations of the RISC-V Berkeley Out-of-Order Machine (BOOM) superscalar core, as listed in Tab. 5.1.

RTL-accurate performance is evaluated for each core configuration for both STV and NTV setups. Power estimations are performed with the Genus Cadence tools by using the 14nm standard cells provided by the FinCACTI library (Shafaei et al., 2014), as it offers implementation designs optimized for both high-performance (STV at 0.8V) and for energy efficiency (NTV at 0.55V). We also use FinCACTI for memory power

Table 5.1 – Explored core configurations. The power and frequency shown are under nominal conditions.

	Parameter	**Rocket**	**SmallBoom**	**MediumBoom**	**LargeBoom**
	Fetch/Decode/Issue width	1	4/1/3	4/2/4	8/3/4
	Issue unit entries	n/a	3×8	3×16	3×24
	Reorder buffer entries	n/a	32	64	96
	Load/store unit entries	n/a	8/8	16/16	24/24
	Integer register file	32	52	80	100
	Floating point register file	n/a	48	64	96
	Power (mW)	16.1	37.7	50.0	81.12
STV	Frequency (GHz)	1.5	1.5	1.5	1.5
	Area (mm^2)	0.10	0.29	0.38	0.59
	Power (mW)	5.5	11.9	15.88	24.5
NTV	Frequency (GHz)	0.22	0.31	0.27	0.22
	Area (mm^2)	0.11	0.3	0.39	0.61

Table 5.2 – Suite of edge-applicable tasks adopted in this work.

Benchmark suite	Tasks
MiBench (GUTHAUS et al., 2001)	AES (enc, dec), CRC32 SHA, Blowfish Susan (corners, edges, smoothing) Qsort, Dijkstra, String search
LOCUS IoT (TAN et al., 2016)	Histogram, A* 2D Convolution Dynamic Time Warping (DTW) Support Vector Machine (SVM)
PolyBench (POUCHET; YUKI, 2011)	Correlation, Covariance Jacobi-1d, Seidel-2d, Cholesky, Ludcmp Gramschmidt, Trisolv, Durbin Floyd-Warshall, Mvt, Gemm 2mm, 3mm Atax, Bicg, Doitgen

estimation.

We use the applications listed in Tab. 5.2 to cover a significant scope of applications on the edge domain. The choice of applications spans across different domains taken from the benchmark suites such as the IoT-domain LOCUS kernels (TAN et al., 2016), the MiBench suite (GUTHAUS et al., 2001), and the PolyBench suite (POUCHET; YUKI, 2011).

To evaluate our approach, we perform chip design space exploration with the following MPSoC configuration cases:

1. *Full STV/NTV* case: All exhaustive chips options that operate fully at STV/NTV;
2. *SNAP* case: All SNAP chip cases are built with the methodology described previously, for each required power limit.

We perform IPS-oriented mappings (oracle mapping) for all cases listed above and report the highest attainable IPS for the best configuration for each required power limit. We assume power limits such that we can cover all possible chip configurations (of lowest to highest power consumption) by varying the power limits in the range [25mW, 800mW], with steps of 10mW.

Table 5.3 – The three evaluated chip cases. Four NTV rocket cores (**4R**) are considered in Config 3. All other cores are at STV.

	#Cores	Configuration	V_{dd} settings	Possible mappings
Config 1	6	2R+2M+2L	STV	90
Config 2	8	7R+1L	STV	8
Config 3	6	**4R**+2L	STV and NTV	15

5.5.2 Results

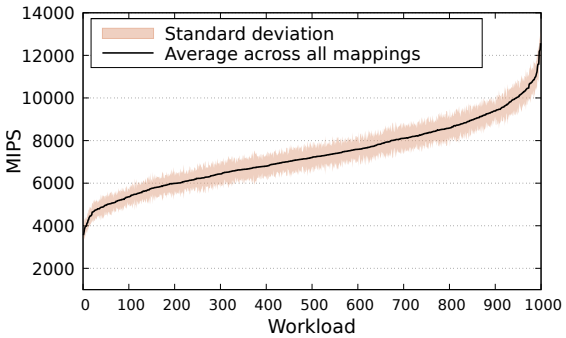
5.5.2.1 Evaluating Application Mappings on Heterogeneous Cores

This section evaluates the MIPS improvements achievable by varying application mappings to heterogeneous cores.

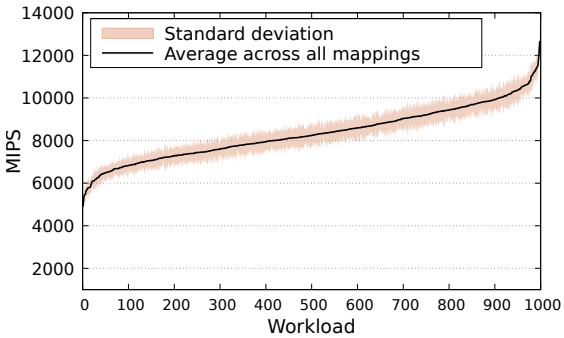
To estimate the potential impacts of application mapping to chips with varying degrees of heterogeneity, we consider the three chip cases shown in Tab. 5.3. For the mapping evaluation procedure, we generate a thousand random workloads composed of random tasks from Tab. 5.2. Each workload consists of as many tasks as the number of cores in each configuration. We then exhaustively run all possible mappings of all workloads in each chip considered.

Figure 5.3 – MIPS distribution for all random workloads when mapped to the heterogeneous chip configurations.

(a) Config 1 (2R+2M+2L)



(b) Config 2 (7R+1L)



(c) Config 3 (**4R**+2L)

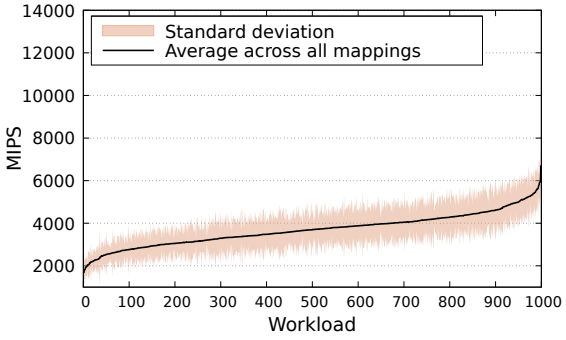
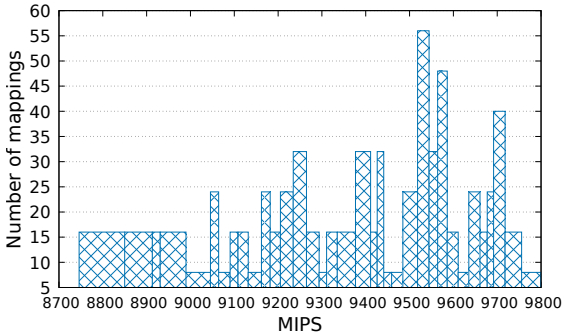
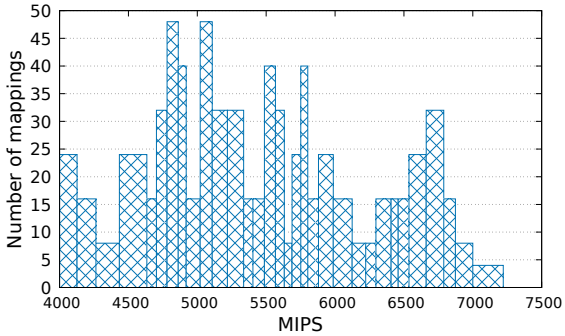


Figure 5.4 – MIPS distribution for the two workloads with the lowest (lowest var) and highest (highest var) degree of MIPS variation across all mappings to heterogeneous chips.

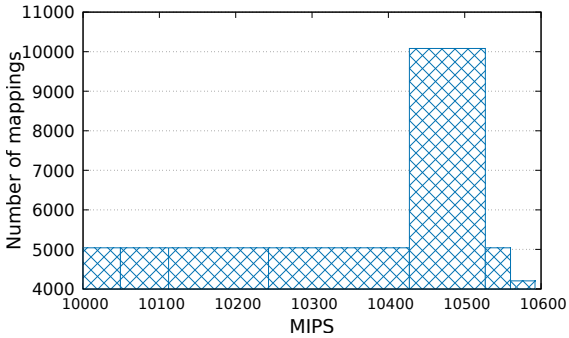
(a) Config 1 (2R+2M+2L) - lowest var (2.7% var)



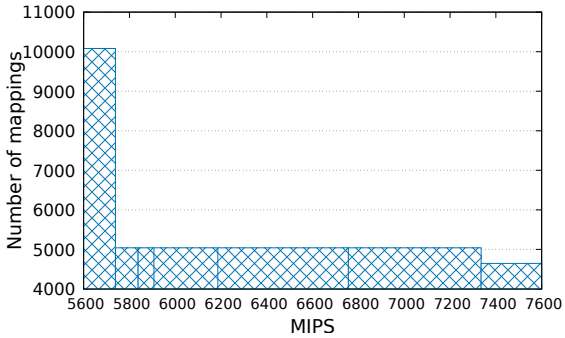
(b) Config 1 (2R+2M+2L) - highest var (13.7% var)



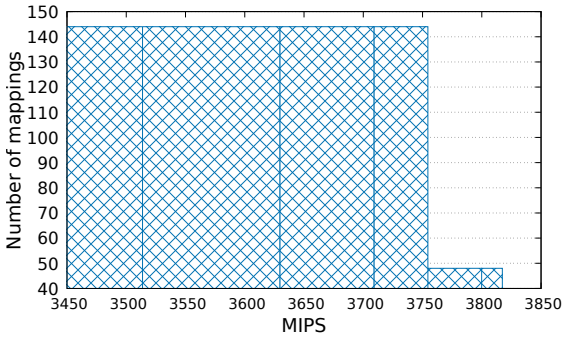
(c) Config 2 (7R+1L) - lowest var (2.1%)



(d) Config 2 (7R+1L) - highest var (10.8%)



(e) Config 3 (**4R**+2L) - lowest var (3.7%)



(f) Config 3 (**4R**+2L) - highest var (53.6%)

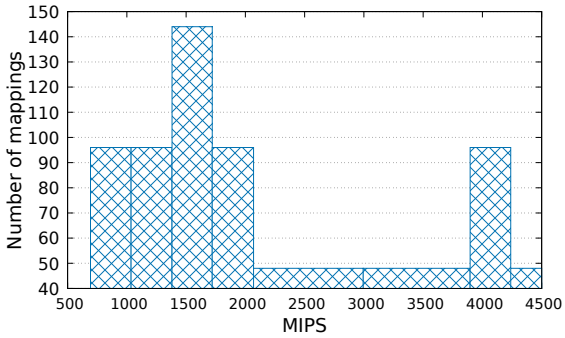


Fig. 5.3 depicts, for each configuration, the average achievable MIPS (in black lines), as well as the MIPS variation (in shaded yellow color) across all mappings for each workload (x-axis). We sort the plots by ascending order of MIPS for each workload for better clarity. All workloads present a significant degree of variation across mappings, and this effect is more pronounced for more heterogeneous architectures (configurations 1 and 3), as explained next.

To characterize the mapping variations for each configuration, we selected the two workloads with the lowest and highest degree of variation across mappings and plotted their corresponding mapping histograms in Fig. 5.4. For both workloads, the plots show that mappings to heterogeneous architectures result in a wider MIPS spread for the most

Table 5.4 – Mapping distribution characterization, for each configuration, for the two workloads with the lowest and highest variation across mappings.

Config	Workload	Average (μ)	Standard Deviation (σ)	Var (%) (σ/μ)	Mapping min	Mapping max	Max/Min
Config 1	Lowest	9391.7	255.6	2.7%	8797.9	9807.2	1.11
(2R+2M+2L)	Highest	5475.8	748.5	13.7%	3982.6	7110.6	1.79
Config 2	Lowest	10339.5	214.0	2.1%	10014.8	10576.2	1.06
(7R+1L)	Highest	6253.5	673.2	10.8%	5651.9	7569.2	1.34
Config 3	Lowest	3659.2	136.4	3.7%	3468.4	3949.9	1.14
(**4R**+2L)	Highest	2390.9	1346.0	56.3%	864.6	4758.8	5.50

heterogeneous configuration, as seen in Figs. 5.4a and 5.4b. Notice that we run all workload permutation cases for each chip. For example, configuration 3 has $6! = 720$ possible mappings, among which only 15 of such mappings are effectively different due to repeating core types. As such, many mappings are placed in the same bin, resulting in a smaller number of larger bins, as shown in the plots.

The MIPS distribution characterization for each chip is shown in Tab. 5.4. The table shows the characterization for the two workloads with the lowest and highest degree of variation. The average and standard deviation of MIPS are shown, as well as the coefficient of variation and the mappings with minimum and maximum MIPS.

Notice that the ratio of best to worst case mappings (in the last column) is significant even for the workloads with the lowest degree of variation, so the potential MIPS improvements that can be achieved by adequately exploring the mappings by smartly matching application characteristics to the most appropriate cores are very compelling. In particular, application mapping plays a more critical role for Config 3. Due to the two voltage islands, this case presents more aggressive mapping-related MIPS variation. The worst-to-best mapping ratio is at least 14% (lowest variation workload) and up to $5.5\times$ (highest variation workload), thus further justifying the importance of proper application mappings when heterogeneous cores are available.

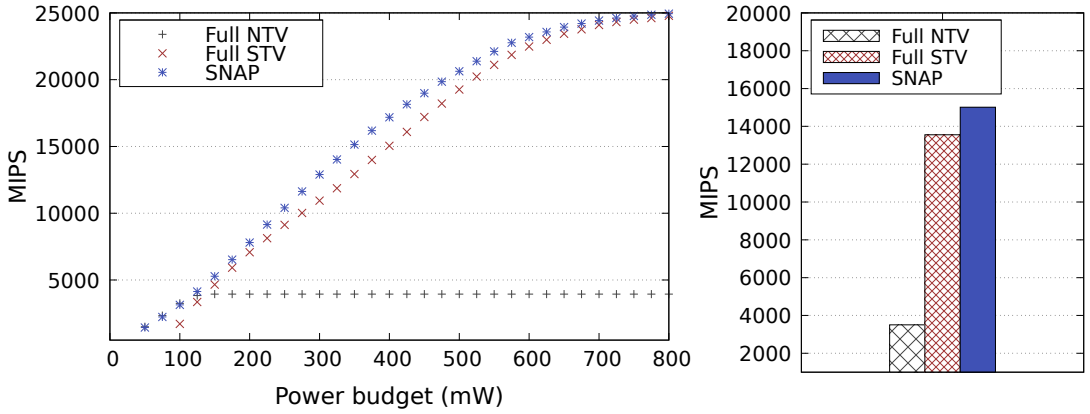
5.5.2.2 Performance Evaluation of SNAP

This section evaluates the achievable MIPS improvements of the *SNAP* over the best possible *Full STV/NTV* architectures.

For the evaluation, we consider power budgets ranging from 25mW up to 800mW and obtain the most performant chip architecture for each design evaluation case (*Full*

Figure 5.5 – MIPS comparison for the different MPSoC composition strategies.

(a) Best average attainable MIPS per budget. (b) Average across all power budgets.



NTV, *Full STV*, and *SNAP*). We then generate one hundred random workloads composed of varying tasks listed in Tab. 5.2, for each chip case and report average results (across all workloads) for the most performant architectures.

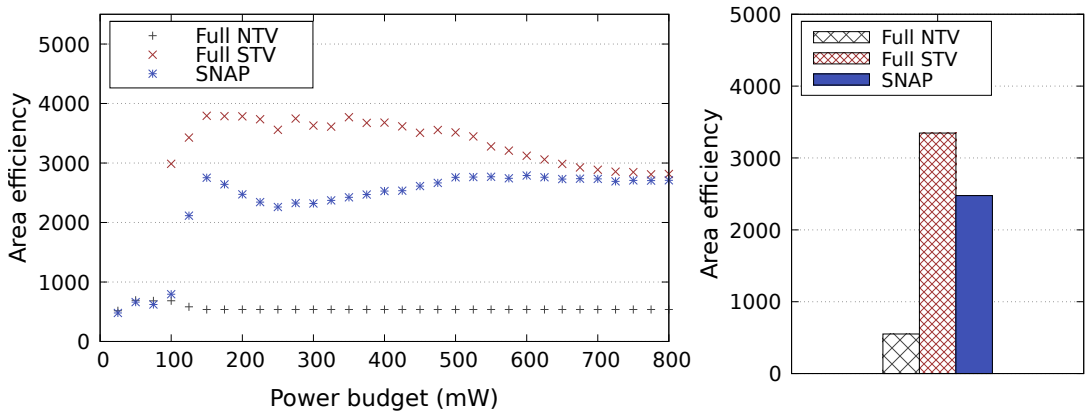
Fig. 5.5 shows the obtained MIPS results for all evaluated cases: *Full STV* (in red), *Full NTV* (in black), and *SNAP* (in blue). We ran our framework for each point in the X-axis (power budget) to determine the best architectures for the three chip pools. As we can observe, for each power budget, the best chip is frequently the SNAP-generated partial-NTV, outperforming both *Full NTV* and *Full STV* cases, except at the extremes of low and high power budgets. For the *Full NTV* version, the most performant configuration is achievable at a budget of around 150mW, so the curve for the *Full NTV* case becomes flat and does not change for higher budgets. For the *Full STV* version, no valid configuration can be achieved for power budgets lower than 100mW.

When the power limit is high enough, the *SNAP* case will optimize all cores to operate at STV. This causes the intersection of the red and blue curves, where the same chip configuration is provided for both STV and SNAP. In between the extremes, a mix of STV and NTV cores provides improved performances, evidencing the advantages of *SNAP*.

In Fig. 5.5b, we report the average performance across all power limits. Performance improvement of *SNAP* is achieved due to the extra throughput that is enabled by the higher number of cores that fit in the power budget when only an efficient subset of NTV cores is employed. On average (i.e., across all power budgets), *SNAP* improves performance upon the best *Full STV* cases on 13.3 percent (up to 83 percent). When compared to the *Full NTV* case, *SNAP* can improve performance up to $6.3\times$ (average of $3.4\times$).

Figure 5.6 – Area efficiency ($MIPS/mm^2$) comparison for the different MPSoC composition strategies.

(a) Best average attainable area efficiency per budget. (b) Average across all power budgets.



5.5.2.3 Area Efficiency Evaluation

NTV designs tend to suffer from poor area efficiency, as operating with high throughput at low voltage settings requires extra hardware resources (e.g., more NTV cores) to compensate for the low frequency of NTV, increasing the chip area. However, SNAP can alleviate such overhead by setting only the necessary portion of the chip to operate at NTV, leaving the remaining cores at STV with higher frequency. Thus, under the same power budget, we can optimize the chip with the partial NTV setting, yielding a more performant system in a smaller area compared to a *Full NTV* chip. Conversely, *Full STV* configurations provide improved performance per area but reduced overall performance within each power budget due to lower instruction throughput offered by the smaller number of available cores compared to *SNAP*.

The area efficiency results are shown in Fig. 5.6, in terms of MIPS per square millimeter ($MIPS/mm^2$), for the three strategies explored (*Full NTV/STV* and *SNAP*). These results were obtained with the same methodology as in the previous section. As expected, NTV cannot improve area efficiency over the *Full STV* version, as it would require operating with higher performance in the same area and power (or with lower performance with less area), which is unattainable with NTV. However, as shown in Fig. 5.6b, the average area efficiency of SNAP (i.e., across all power budgets and for each scenario) is improved over the *Full NTV* case. The gains range from 16 percent and up to $5.2\times$ (average improvement of $3.9\times$). Thus, under a power limit, although *SNAP* has poorer efficiency when compared to the *Full STV* designs, we can alleviate the poor efficiency of traditional *Full NTV* designs.

6 A VARIATION-AWARE METHODOLOGY FOR IMPROVED PROCESSOR DESIGNS FOR THE EDGE COMPUTING DOMAIN

This chapter provides details on our proposed methodology for variation-aware chip optimization. The general optimization goal is **to provide efficient chip configurations that (1) maximize the system's instruction throughput under a power limit while also (2) maintaining minimum yield requirements of chips subject to process variation scenarios.**

In the following section, We provide a case study to justify our effort to tackle variations. The following sections then provide the implementation details of our proposed methodology.

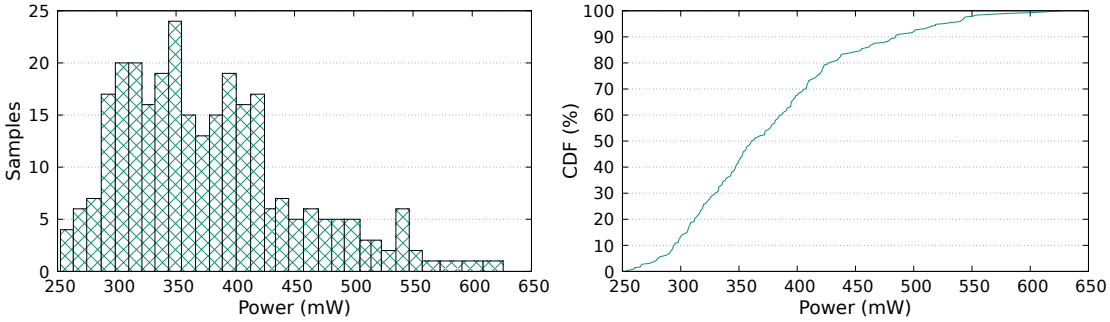
6.1 Motivational Analysis

NTV-induced process variations can lead to increased delay and leakage, causing frequency fluctuations and unwanted additional power dissipation, thus limiting overall energy efficiency. This issue is particularly problematic for low-power edge devices, such as battery-powered processors, that must adhere to strict power dissipation limits. To address such challenges, it is crucial to properly explore design margins in the low-power domain to achieve high performance and energy efficiency while maintaining acceptable yield levels under the constraints of process variation.

As an example of a variation-afflicted design, Fig. 6.1 depicts the power distribution for several samples of a heterogeneous chip. Notice the distribution is normal, so we plot its histogram closely resembling a Probability Density Function (PDF) (Fig. 6.1a)

Figure 6.1 – Power distribution of a heterogeneous chip configuration (4 small + 4 big cores). The samples were generated as explained in section 6.5.1.

(a) PDF of the power samples. (b) CDF of the power samples.



and its associated Cumulative Distribution Function (CDF) (Fig. 6.1b).

The chip in Fig. 6.1 is composed of 4 little + 4 big cores, a configuration resembling the Arm's DynamIQ setups (ARM, 2021) such as the Snapdragon 865 and Exynos 990 (Triggs, Robert, 2021). The chip's ideal/nominal peak power dissipation (i.e., with no variation effects) is $\approx 308mW$. However, variation shifts power according to the Probability Density Function (PDF) distribution in Fig. 6.1a. The power samples' Cumulative Distribution Function (CDF) is shown in Fig. 6.1b. If the chip's peak power constraint is 400mW, for example, then $\approx 32\%$ of the chips will exceed the power limit ($cdf_{power}(400mW) \approx 68\%$). This issue can severely degrade yield for battery-powered processors with strict power budgets.

6.2 Variation-Aware Proposal

Our approach comprises optimization methodologies starting in the design stage, followed by a post-design and variation-aware frequency adaptation proposal. The optimization goal is to maximize chip-level instruction throughput under two restrictions: (1) a user-defined power limit; and (2) the process variation constraints that affect performance and power. Additionally, notice that variation-induced power overheads affect the yield because such variations may result in power dissipation beyond the desired power limit. Consequently, to assure minimum yield maintenance while maximizing performance in a variation scenario, we propose a methodological workflow comprising steps at varying system layers that allows for better exploration of the predicted process variation margins.

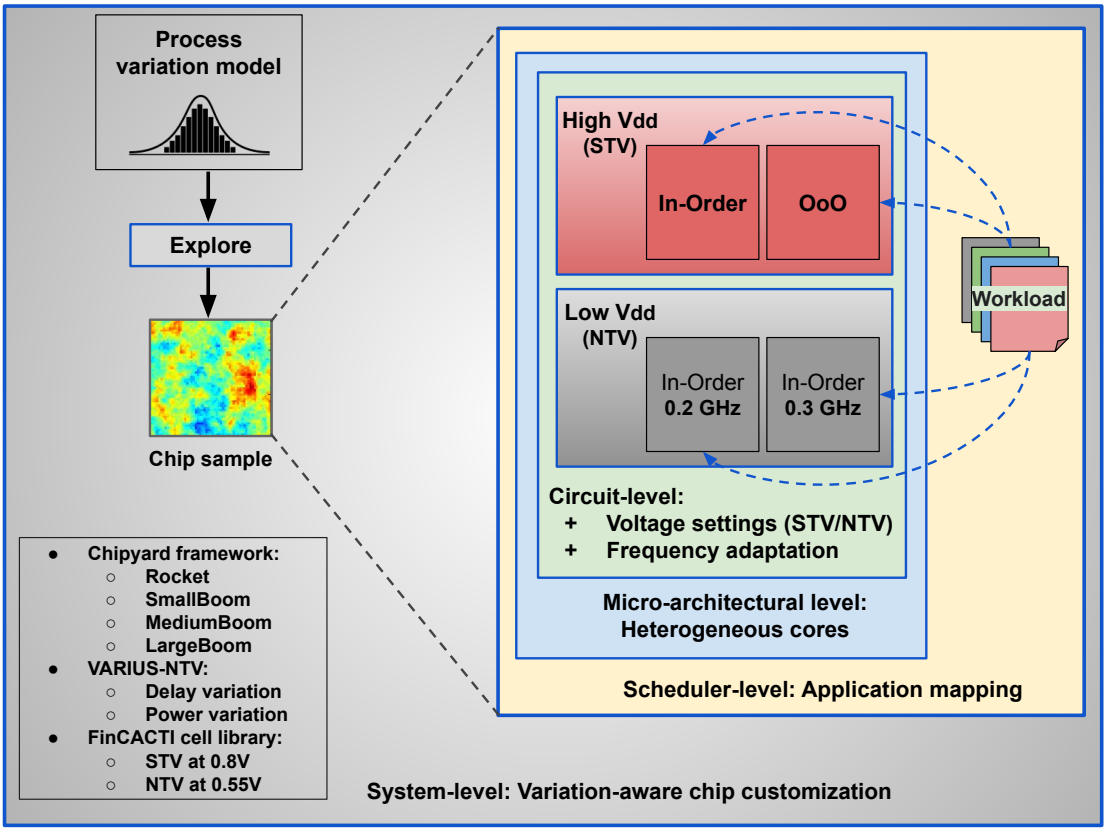
The whole approach spans multiple layers of the chip design, from the circuit level up to the level of application mapping. An abstract view of this work is shown in Fig. 6.2. Each optimization layer (as shown in the figure) can be resumed as follows:

1. *Circuit level*: Here, we propose implementations of chips with two voltage settings to accommodate both high V_{dd} (STV) and low V_{dd} (NTV) cores in the same die. Additionally, we perform post-design per-core frequency adaptation to address variation-induced delay and power variations;
2. *Micro-architecture level*: We propose an efficient chip composition algorithm that provides optimized heterogeneous systems considering four different core micro-architectures, including scalar in-order and more complex superscalar out-of-order

(OoO) cores. The core designs are taken from the Chipyard framework (Amid et al., 2020). Chipyard provides customized core designs suitable for this research. This stage results in efficient chip compositions with mixes of different core micro-architectures as well as the proper voltage settings of each core;

3. *Scheduler level*: After chip design, we perform proper application mappings to heterogeneous cores aiming to maximize overall/chip-level instruction throughput by better matching application properties to the most suitable core. We model the mapping problem as a *maximum weighted perfect matching* assignment problem to provide mappings with maximized chip-level instruction throughput;
4. *System level*: The optimizations across all layers provide optimized chip configurations with improved system-wide instruction throughput under restricted power limits and process variation scenarios. Moreover, by carefully selecting the most suitable chip architectures under the power and variation constraints and performing proper post-design frequency adaptation, we meet minimum yield constraints that can otherwise be violated if the variation margins are not efficiently harnessed during the design and post-design stages.

Figure 6.2 – System layers explored in this thesis.



6.3 Approach Overview

This section provides preliminary background information on the scope of this work. We start by highlighting the overall flow of our optimization proposal (Sec. 6.3.1) and provide background information on (1) the abstract chip architectural model (Sec. 6.3.2); (2) how we model and address process variation (Sec. 6.3.3), and (3) the adopted application mapping methodology to explore heterogeneity during runtime (Sec. 5.3).

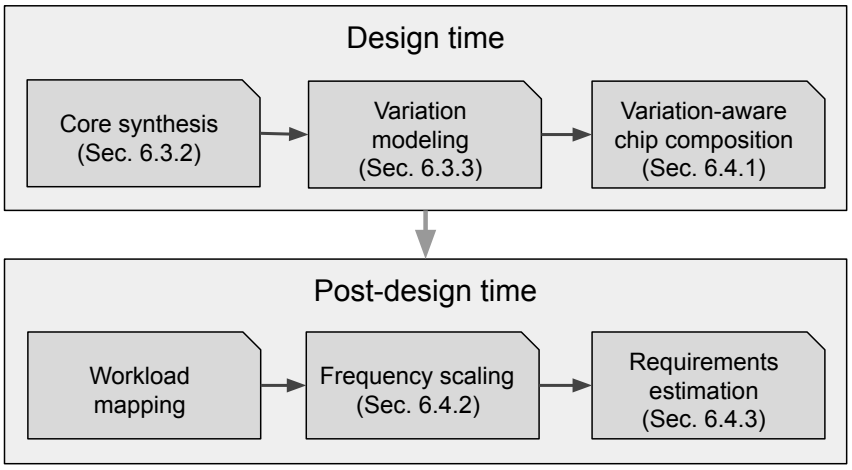
6.3.1 Proposed Flow of Optimization

An abstract view of our workflow is shown in Fig. 6.3. We start by performing logic synthesis of each core design. We then apply process variation statistical models in the designs and perform chip configuration exploration under the assumption of variation that maximizes performance under power limits. In the post-design phase (test phase), we perform workload mapping and apply an efficient variation-aware frequency scaling to further improve the power/performance of the configurations selected during the design stage. The design-time chip customization approach and the proposal for post-design frequency scaling will be further detailed in Secs. 6.4.1 and 6.4.2.

6.3.2 Chip Architecture Exploration Scope

In this work, we consider variation-aware architectural chip exploration with both STV and NTV voltage operation points in the same chip. An abstract architectural-level

Figure 6.3 – Abstract workflow of this thesis.



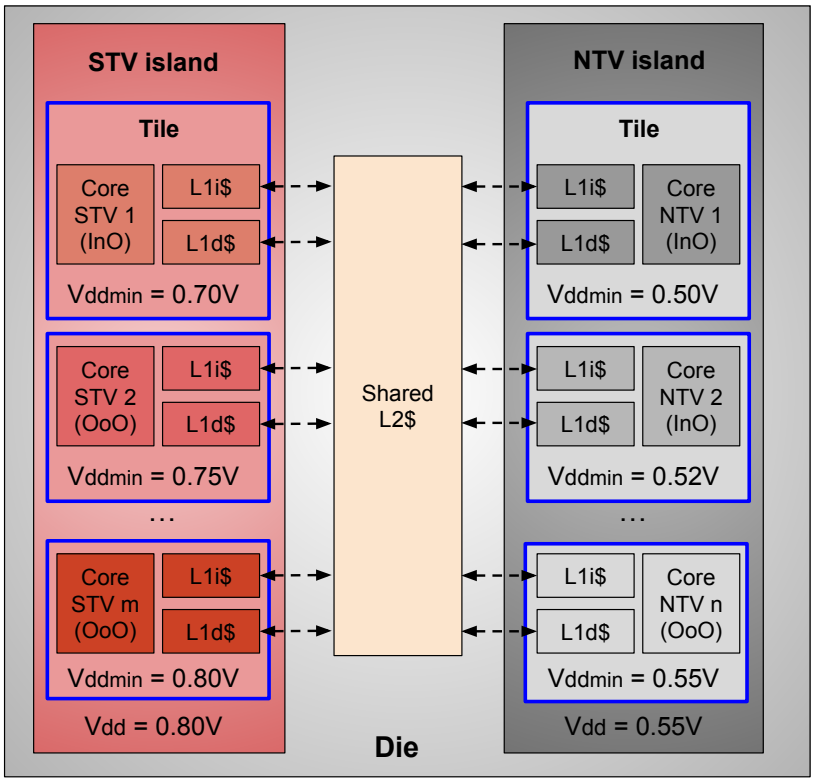
view of the proposed chip is shown in Fig. 6.4. This approach is similar to the one presented in the last chapter. However, we complement and improve the previous approach in the following ways: 1) we introduce parameter variations models in the cores, 2) at design time, we augment the SNAP exploration with extra variation information to improve the chip composition methodology, and 3) at post-design time, we propose a frequency adaptation mechanism to improve performance and/or yield in a power limit and variation scope. The approach is termed Variation-Aware SNAP (or VA-SNAP).

6.3.3 Modeling and Addressing Parameter Variations

Parameter variations caused by (mostly uncontrollable) chip production irregularities, such as from lithography imperfections (systematic) and random effects, result in physical mismatches from the intended and nominal goals. Among other parameters, variations can affect the transistor's threshold voltage (V_{th}) and the effective channel length ($Leff$), according to Eq. 6.1. $V_{th}0$ and $L_{eff}0$ denote nominal values, which are shifted by random ($rand$) and systematic (sys) variation components.

Such variations cause transistor delay variation as well as leakage current varia-

Figure 6.4 – Architecture-level view of the chip subject to process variations.



tion. The former manifests in the form of variability in attainable frequency (if the critical path is affected by variation), and the latter manifests in the form of excessive static power dissipation due to variation-induced extra leakage current.

$$\begin{aligned} L_{eff} &= L_{eff}0 + \Delta L_{eff} = L_{eff}0 + \Delta L_{eff,rand} + \Delta L_{eff,sys} \\ V_{th} &= V_{th}0 + \Delta V_{th} = V_{th}0 + \Delta V_{th,rand} + \Delta V_{th,sys} \end{aligned} \tag{6.1}$$

In this work, we model variations in both STV and NTV designs. However, the same amount of V_{th} and $Leff$ variations translate in a more pronounced delay and leakage variation at NTV than STV. This can be seen, for instance, by inspecting Eq. 2.5. The term $V_{gs} - V_{th}$ suggests that at low V_{gs}, variations in V_{th} have a more pronounced effect in the $delay_{gate}$ distribution. If not effectively addressed, both types of variations, delay, and leakage, affect microprocessor design in at least the three following ways:

1. **Timing failures due to delay variation:** Timing failures (and memory access timing failures) arise when delay variation affects a logic path making it slower than the microprocessor's clock period (Ernst et al., 2004).
2. **SRAM stability**: Variation can cause hold and write SRAM failures if not addressed. Achieving proper SRAM operation at NTV is more difficult because SRAMs usually rely on very meticulous and precise sizing of transistors. As a consequence, SRAMs tend to be more sensitive to variation, and assuring the read and write stability of such designs is more challenging than other logic types;
3. **Degraded yield due to excessive power dissipation:** Excessive leakage current can occur due to variations in the channel length (Borkar et al., 2003). This is a conundrum as microprocessors must adhere to a particular power envelope due to the design-specific TDP or, for instance, to improve battery-powered devices' lifetime reliability or energy efficiency. If not addressed, devices that consume excessive power must be discarded, thus degrading yield.

This work adopts the parameter variation models taken from the VARIUS-NTV framework, described in (Karpuzcu et al., 2012). VARIUS-NTV models both V_{th} and L_{eff} variations that affect transistor delay and static power dissipation. Both spatially correlated systematic and non-correlated random variations are modeled.

Our methodology consists in first performing logic synthesis of all explored core types. We then extract the delay and static power variation maps provided by VARIUS-

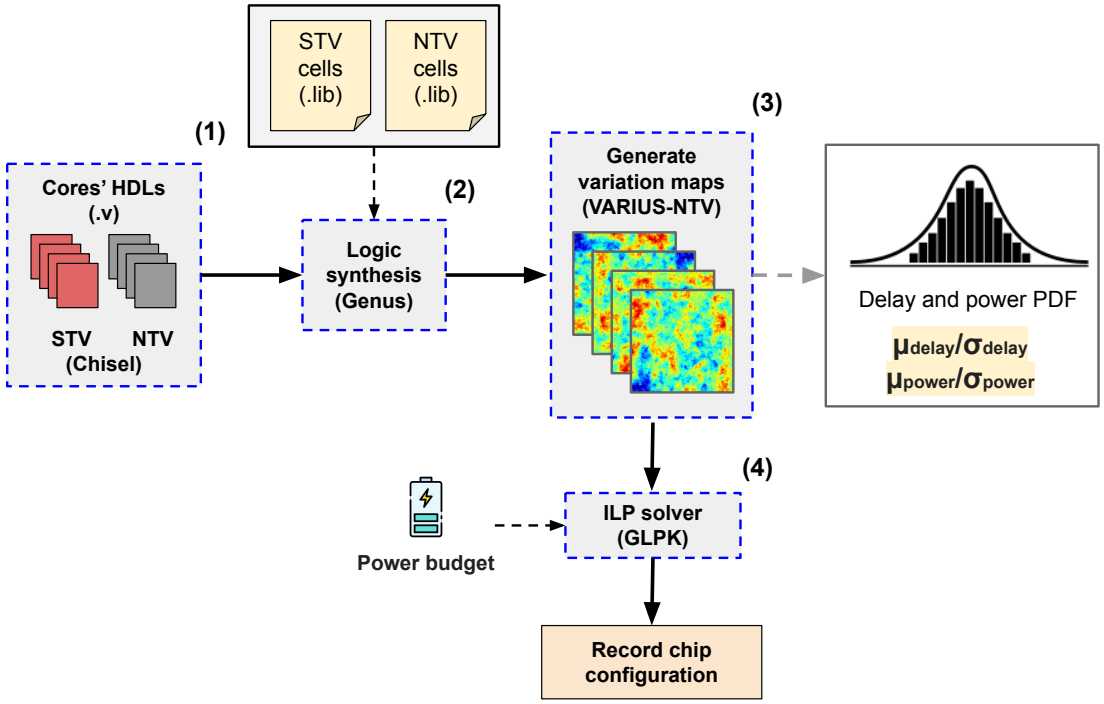
NTV with standard floorplanning provided by VARIUS-NTV. Lastly, we generate the power and delay distributions for each core by scaling the obtained synthesis data according to the variation maps provided by VARIUS-NTV.

Recall that we adopt no more than two voltage islands in our design. This provides higher energy efficiency when compared to multiple voltage settings, primarily due to the limited power efficiency of having multiple voltage regulators (Karpuzcu et al., 2013). This way, we do not address delay variation by performing per-core V_{dd} adaptation. Instead, variation is addressed by (1) adopting 8-transistor SRAM cells for improved access stability; (2) assuring a single safe voltage level (per voltage island) that sustains safe functionality of all tiles in each island, and (3) performing per-core frequency adaptation for the given voltage levels and delay variation constraints. These three steps are detailed as follows:

1. *Adopting 8-transistor SRAM cells*: We consider 8-transistor SRAM cells for the NTV designs. This is the variation model implemented by VARIUS-NTV, and works more reliably at low voltage operation because it protects against read upsets by decoupling the read and write access transistors. This contrasts with the more traditional 6-transistor SRAM designs that are more sensitive to variation. For NTV designs, we estimate power and area for this cell model from the FinCACTI model and use VARIUS-NTV to estimate parameter variations;
2. *Determining the safe voltage level*: This step is delegated to the VARIUS-NTV framework and is not a contribution of this thesis. However, we explain this step to improve text clarity. VARIUS-NTV computes safe voltage levels to avoid hold and write stability failures. This step is performed analytically by evaluating each SRAM block in the chip and choosing the highest minimum voltage level among all blocks. Assuring proper SRAM functionality at NTV guarantees the safe operation of the other logic types (combinational and flip-flop-based sequential logic), because SRAMs are the bottleneck logic type to optimize at NTV;
3. *Determining the frequency*: After the minimum voltage level is computed, VARIUS-NTV proceeds to perform static timing analysis in the logic distribution to estimate transistor delays with the voltage computed in the previous step. This step provides delay variation maps and the per-core (variation-afflicted) critical path.

We address timing failures by applying proper frequency adaptation, after the chip design, by plugging the delay variation maps provided by VARIUS-NTV into each of the

Figure 6.5 – Design time chip exploration workflow.



chip configurations we evaluate. The frequency adaptation is made statically after the chip design. This work part is detailed in Sec. 6.4.2.

6.4 Variation-Aware Design- and Post-Design time Optimization

This section provides details on the elaboration of our methodology. We start by explaining the design-time chip customization strategy (Sec. 6.4.1), followed by the elaboration on the post-design frequency scaling mechanism (Sec. 6.4.2). Lastly, we elaborate on the high-level algorithm encompassing both design- and post-design phases (Sec. 6.4.3). The design-time and post-design-time workflows are outlined in Figs. 6.5 and 6.6, respectively. We will refer to the figures in more detail in the next sections.

6.4.1 Design-time and Variation-Aware Chip Customization

The overview of the design time optimization workflow is shown in Fig. 6.5. The optimization goal is to find optimized configurations to meet the criteria shown in Eq. 6.2. For the user-defined power budget (P_b) and a set of configurations ($configs$), we seek the MPSoC configuration ($target\_config$) with the highest IPS that provides at least the minimum required yield $Yield_{min}$. For instance, for a minimum yield of 95%,

we require configurations with $cdf_{power}(P_b) \geq 95\%$. Notice that $1 - cdf_{power}(P_b)$ represents the fraction of chip samples that exceed the power limit. Our goal is to identify the heterogeneous configuration with the highest performance and lowest probability of violating the power budget, considering the distributions of performance and power resulting from variation phenomena.

$$TargetConfig(P_b, Yield_{min}) = \underset{conf \in configs}{\arg\max} \{IPS_{conf} | Power_{conf} \leq P_b, \\ cdf_{power}(P_b, conf) \geq Yield_{min}\} \tag{6.2}$$

Recall that our exploration space considers RISC-V heterogeneous cores with $\sum_{i=2}^{16} \binom{i+4-1}{4} = 4840$ full STV and full NTV chips. The VA-SNAP/mixed configurations, however, result in a much larger design space, with a total of $\sum_{i=2}^{16} \binom{i+8-1}{8} = 735,462$ possibilities. In our sampling-based exploration, we generate 288 chip samples per configuration (the variation maps provided by the VARIUS-NTV framework), amounting to over $735,462 \times 288 \approx 211$ million chip variation samples in total, making exhaustive optimization unfeasible (especially due to the frequency scaling solver presented in the next section).

To make the VA-SNAP approach feasible, we improve our ILP approach to efficiently compose optimized MPSoC configurations under the assumption of process variation.

$$\underset{\forall\ core\ types\ c}{\text{maximize}} \quad \sum_{c} \left[(\mu_{IPS,c} + \eta_1 \times \sigma_{IPS,c}) \times Count_c\right], \forall \eta_1 \in [-3,3] \tag{6.3a}$$

$$\text{subject to}$$

$$\mu_{pwr,L2} + \sum_{c} \left[(\mu_{pwr,c} + \eta_2 \times \sigma_{pwr,c}) \times Count_c\right] \leq P_b,\ \forall \eta_2 \in [-3,3], \tag{6.3b}$$

$$2 \leq \sum_{c} Count_c \leq 16 \tag{6.3c}$$

The goal is to maximize the IPS performance (objective) under power budget limits (constraint). The objective and constraints are depicted in Eqs. 6.3a, 6.3b and 6.3c. The inputs are the power budget limit (P_b), the average core IPS and power ($\mu_{ips,c}$, $\mu_{pwr,c}$) of each core type c, the second-level cache average power ($\mu_{pwr,L2}$), as well as the standard deviation information that captures both core IPS and power of chips subject to variations

($\sigma_{ips,c}$, $\sigma_{pwr,c}$) of each core type c. The ILP method aims to select the optimal set of cores (outputs are the types and number of each core type - $Count_c$) to maximize the IPS performance while not violating the given power budget (P_b), as is illustrated in Eq.6.3a (budget constraint in Eq. 6.3b). We also constrain the number of cores to the range [2,16] (constraint in Eq. 6.3c).

Our exploration starts by performing logic synthesis for each core type (step 1 in Fig. 6.5) and generating delay and power distributions (variation maps) for each core configuration (step 2). As power and delay variations are normally distributed, we explore variation margins by conservatively assuming that delay and power vary in the range $[\mu - 3\sigma, \mu + 3\sigma]$. This interval accounts for over 99% of all samples in the distribution.

To find optimized configurations for a required and fixed power limit, we iterate a nested loop over a range of different variation margins (the multipliers η_1, $\eta_2 \in [-3, 3]$ at steps of 0.5), shown in Alg. 1. For each loop iteration (lines 5-6), we run the ILP approach (line 7) with the inputs η_1/η_2 variation factors, the power budget and per-core delay, and power μ/σ variation parameters from their respective PDFs (Probability Density Function), and record the solver's optimized configuration for the required power limit (line 8). This results in no more than 169 configurations per budget (we exclude possible configuration repetitions generated by the solver). The ILP phase is shown in Fig. 6.5, step 4.

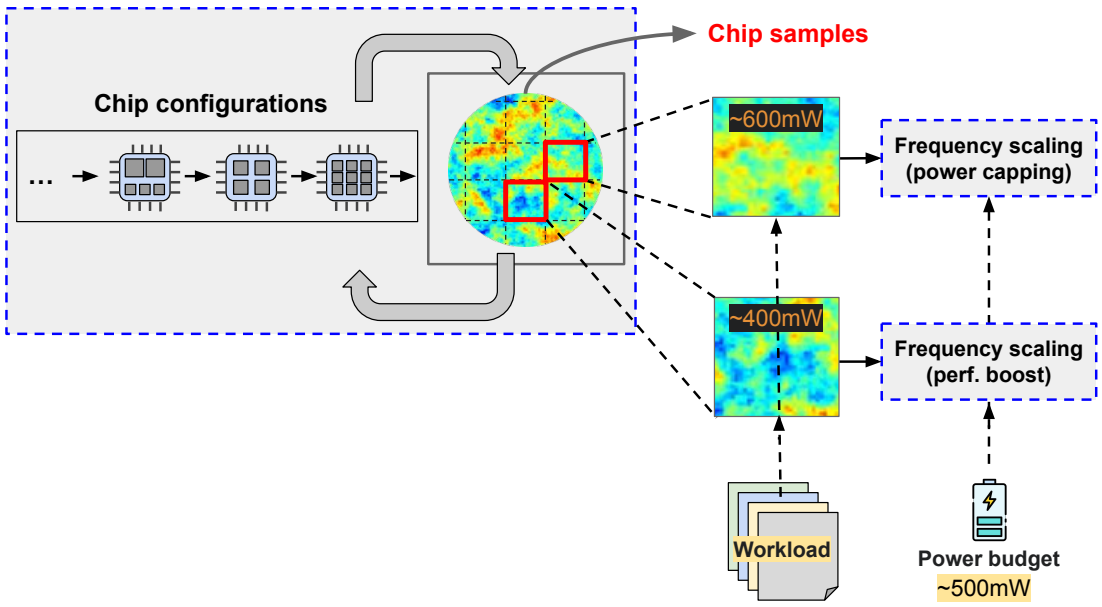
We repeat the previous step (4) over a range of varying power budgets in the interval [50mW, 1000mW], with steps of 10mW. In the end, the ILP process results in a pool of VA-SNAP MPSoC configurations for all required power limits. In total, this approach selects around 15K candidate configurations in the explored power budget range (exclud-

Algorithm 1: ILP algorithm for MPSoC composition

Input: P_b - the power budget
Input: PDF_{delay} - Per core delay Probability Density Function
Input: PDF_{power} - Per core power Probability Density Function
Output: VA-SNAP MPSoC configuration

1 $\mu_{IPS} \leftarrow ExtractAverage(PDF_{delay})$;
2 $\sigma_{IPS} \leftarrow ExtractStandardDeviation(PDF_{delay})$;
3 $\mu_{power} \leftarrow ExtractAverage(PDF_{power})$;
4 $\sigma_{power} \leftarrow ExtractStandardDeviation(PDF_{power})$;
5 **foreach** $\eta 1$ *in range[-3,3]* **do**
6 **foreach** $\eta 2$ *in range[-3,3]* **do**
7 Run ILP in Eq. 6.3a with inputs $\eta 1$, $\eta 2$, P_b, μ_{IPS}, σ_{IPS}, μ_{power}, σ_{power};
8 Record the solved configuration (each $Count_c$);

Figure 6.6 – The goals of frequency scaling.



ing any repetitions). Finally, the configuration pool is fed to the sampling strategy that iterates over each MPSoC configuration in the pool. In this phase, while optimizing for each fixed budget P_b, multiple variation samples are generated for each configuration. The frequency scaling strategy (presented next) is applied to each sample to meet the power limit. We then record each of the configurations' IPS performance and $cdf_{power}(P_b)$ for the target power limit (step 5). In the end, we search for the highest-performing configuration that guarantees a minimum required yield for the desired power limit (extracted from $cdf_{power}(P_b)$).

6.4.2 Post-Design and Variation-Aware Frequency Scaling

We complement our design-time approach with an efficient post-design frequency scaling mechanism to cope with unwanted power variations after the chip design. Fig. 6.6 outlines the goals of our proposed frequency scaling strategy. The frequency adaptation is applicable individually, per chip sample, aiming to either (1) reduce dynamic power dissipation (power capping) if the chip's peak power dissipation is higher than the required budget or otherwise (2) improve performance, if possible. The two goals are described as follows:

1. **Frequency scaling for power capping**: After fabrication, the individual cores vary in delay and power dissipation. If the core's peak power exceeds the desired power

limits, we perform frequency scaling to reduce the core's dynamic power. In this sense, the frequency scaling strategy allows the designer to relax the prefabrication estimated power variation margins. For a given power limit, relaxing the design-time predicted power variation allows the fabrication of more aggressive MPSoC configurations, with higher performance, for the desired power limits;

2. **Frequency scaling to improve performance**: If the chip's peak power dissipation is below the desired power limit, we can efficiently increase the core's frequency in the cases where we can take advantage of (beneficial) delay variation that allows individual cores to operate faster than the designer's target frequency goal. Notice that delay variation follows a normal distribution, where variation samples fluctuate around the intended frequency goal. This essentially means that around 50% of the chip sample designs in the delay distribution have delay below the intended goal, thus allowing for performance boost of such cases.

$$\text{maximize} \quad \sum_{core=1}^{N} (IPC_{core} \times F_{core}) \tag{6.4a}$$

$$\text{subject to}$$

$$P_{L2} + \sum_{core=1}^{N} Psta_{core} + \sum_{core=1}^{N} Pdyn_{core} \leq P_B, \tag{6.4b}$$

$$50MHz \leq F_{core} \leq F_{max,variation}, \forall cores \in [1, N] \tag{6.4c}$$

For each chip sample, the total power dissipation includes the cores' dynamic ($Pdyn_{core}$) and static ($Psta_{core}$) powers as well as the second-level cache total power (P_{L2}). Namely, the chip's total power is modeled as $P_{chip} = P_{L2} + \sum (Psta_{core} + Pdyn_{core})$, where $Pdyn_{core} \propto F_{core}$; so scaling the frequency brings associated changes the cores' dynamic power dissipation.

The proposed frequency scaling methodology is implemented as a Linear Programming (LP) approach, shown in Eq. 6.4a, which aims to keep total power dissipation under the budget P_B while maximizing the system's overall IPS as much as possible. For that, the LP aims at finding the optimal set of core frequencies (F_{core}, for all N cores in the MPSoC) such that the overall MPSoCs IPS is maximized (notice that $IPS_{core} = IPC_{core} \times F_{core}$) while not exceeding the desired power limit (Eq. 6.4b).

Neither P_{L2} nor $Psta_{core}$ are affected by frequency scaling. Also, to keep the

Algorithm 2: Variation-aware exploration algorithm

Input: P_b - the power budget
Input: $Yield_{min}$ - the minimum required yield
Input: $workload$ - a set of applications
Output: Best MPSoC configuration for the power restriction and minimum yield

```
/* Generates the list of MPSoC configurations.  Two cases:                  */
/* 1) Full STV/NTV configurations are created exhaustively.                 */
/* 2) SNAP configurations are created with the method in
   brownSec.  6.4.1, Alg.  1                                                */
ConfigPool ← GenerateConfigsList(Pb, Full STV/NTV or SNAP);
currentConfig ← 0;
ipsBest ← 0;
/* Iterates over each MPSoC configuration                                   */
foreach configuration config in ConfigPool do
    chipSamples ← GenerateVariationSamples(config);
    countPass ← 0;
    μips ← 0;
    foreach MPSoC varSample in chipSamples do
        varSample.mapWorkload(workload);
        /* Scale frequency with the method in Sec.  6.4.2, Eq.  6.4a        */
        if varSample.LpTuneFrequencyOpt(Pb) == TRUE then
            countPass ← countPass + 1;
            μips ← μips + varSample.ips;
        end
        else
            /* Sample failed to meet the power budget.                      */
        end
    end
    μips ← μips/countPass;
    yield ← countPass/chipSamples.items;
    if yield ≥ Yield_min then
        /* Keeps the best configuration for this budget                     */
        if μips > ipsBest then
            ipsBest ← μips;
            configBest ← currentConfig;
        end
    end
    else
        /* Configuration cannot meet the minimum required yield for this
           budget.                                                          */
    end
    currentConfig ← currentConfig + 1;
end
return configBest;
```

design exploration feasible, we assume that the IPC is not affected by frequency. While frequency could affect IPC to some extent, this simplification makes the maximization problem linear and reduces the complexity of our design space while still providing efficient frequency solutions. Finally, notice that in the restriction in Eq. 6.4c, the core's frequency is allowed to be boosted up to the maximum frequency allowed by variation ($F_{max,variation}$), which can be either below or above the designer's target frequency goal.

6.4.3 Putting It All Together: Variation-Aware Exploration Algorithm

The complete variation-aware exploration algorithm is shown in Alg. 2. The algorithm works in a per-workload fashion. Workloads are defined as sets of tasks, as

described in section 6.5.1.

First (line 1), we generate the pool of MPSoC configurations we want to explore. For the full STV and full NTV versions (4840 configurations each), we exhaustively list all possible configurations. For the VA-SNAP MPSoCs with mixes of STV and NTV cores, however, we use the ILP method described in subsection 6.4.1 to list the candidate configurations we want to explore (around 15k configurations). Secondly, we generate the variation samples for each configuration (line 5) and map the desired workload to each sample (line 9), determining the cores' IPCs and power dissipation used in the frequency scaling phase. We then perform the frequency scaling optimization for the sample targeting a power limit P_b (line 10), which returns $TRUE$ in case there is a set of possible valid frequencies such that the current MPSoC sample consumes no more than the required power limit p_b; otherwise the optimization fails, and the sample is discarded.

We keep a sample counter $countPass$ to count the number of samples that consume power less than or equal to the power limit, i.e., the ones that are not discarded (line 11). This counter determines the average IPS of all such samples (line 17), as well as the yield (line 18) for the required power limit, which determines $cdf_{power}(pb)$. Next, if the yield is at least equal to the minimum required (line 19), we select the highest-performing configuration, i.e., the configuration of maximum IPS such that $cdf_{power}(pb, config) \geq Yield_{min}$ (lines 20-22), achieving the optimization goal of Eq. 6.2.

6.5 The Proposal's Evaluation

This section presents the results of our variation-aware chip composition and the proposed frequency scaling methodology.

6.5.1 Experimental Methodology

We conducted our experiments with the core implementations provided by the Chipyard open-source framework (Amid et al., 2020). The adopted core configurations are the RISC-V Rocket (in-order) core as well as three configurations of the RISC-V Berkeley Out-of-Order Machine (BOOM) superscalar core, as listed in Tab. 6.1.

RTL-accurate performance is evaluated for each core configuration for both STV and NTV setups. Power estimations are performed with the Genus Cadence tools by

Table 6.1 – Explored processor configurations. The power and frequency shown are under nominal conditions.

	Parameter	Rocket	SmallBoom	MediumBoom	LargeBoom
	Fetch/Decode/Issue width	1	4/1/3	4/2/4	8/3/4
	Issue unit entries	n/a	3×8	3×16	3×24
	Reorder buffer entries	n/a	32	64	96
	Load/store unit entries	n/a	8/8	16/16	24/24
	Integer register file	32	52	80	100
	Floating point register file	n/a	48	64	96
	Power (mW)	16.1	37.7	50.0	81.12
STV	Frequency (GHz)	1.5	1.5	1.5	1.5
	Area (mm^2)	0.10	0.29	0.38	0.59
	Power (mW)	5.5	11.9	15.88	24.5
NTV	Frequency (GHz)	0.22	0.31	0.27	0.22
	Area (mm^2)	0.11	0.3	0.39	0.61

using the 14nm standard cells provided by the FinCACTI library (Shafaei et al., 2014), as it offers implementation designs optimized for both high-performance (STV at 0.8V) and for energy efficiency (NTV at 0.55V). We also use FinCACTI for memory power estimation.

To cover a significant scope of applications on the edge domain, we use the applications listed in Tab. 6.2. We categorize the applications spanning across different domains, as listed in the table. The applications Support Vector Machine (SVM), A*, Dynamic Time Wrapping (DTW), AES, and Histogram are taken from the IoT-domain LOCUS kernels (TAN et al., 2016). Dijkstra, SHA, CRC32, and Susan are taken from the MiBench suite (GUTHAUS et al., 2001), and the linear algebra and data mining applications are taken from the PolyBench suite (POUCHET; YUKI, 2011).

To evaluate our approach, we extract within-die variation samples from the VARIUS-NTV framework and perform the exploration algorithm with the following MPSoC configuration cases:

1. *Full STV/NTV* case: The variation-aware methodology is applied in the two design pools consisting of MPSoCs operating fully at STV/NTV. Notice that both *Full STV/NTV* cases are always "variation-aware" as we explore all possible configurations for both cases;
2. *SNAP* (variation unaware) case: This is the case where variation is not considered in the SNAP methodology (Chapter 5), i.e., the ILP method for MPSoC formation considers only nominal IPS and power, with no variation margin assumptions;

3. *VA-SNAP* (variation aware) case: The VA-SNAP methodology as described in section 6.4.

For all cases listed above, we perform the sampling-based search both with frequency scaling (+FS) and without it (no FS) and report the average attainable IPS (i.e., average across all samples) for the best configuration (i.e, of highest average IPS) provided by the sampling method, assuring minimum yield requirement of 95% for each required power limit. We assume power limits such that we can cover all possible chip configurations (of average lowest to average highest power consumption, assuming process variation) by varying the power limits in the range [25mW, 1000mW], with steps of 10mW. We consider all scenarios in Tab. 6.2, providing a wide distribution in terms of MIPS across cores and tasks (see Fig. A.4 in appendix A.2). The results report the average optimized IPS across all application scenarios.

In this section, we present the obtained performance results for the optimized configurations for the three MPSoC configuration cases listed in section 6.5.1. For each power budget P_b, we search for the best configuration with a minimum yield of 95% (i.e., with $cdf_{power}(P_b) \geq 95\%$). However, we evaluate achievable performance for different yield requirements in subsection 6.5.2.4.

Table 6.2 – Application scenarios explored in this work.

Scenario	App domains	Applications
1	Image processing Machine learning Graphs Signal processing	Susan corners SVM A*, Dijkstra, Floyd–Warshal DTW
2	Security Data mining Data integrity	Blowfish Correlation, Covariance SHA, CRC32
3	Misc Algebra Graphs	String search LU, Cholesky, Gramschmidt A*, Dijkstra, Floyd–Warshal
4	Misc Image processing Security	Qsort Susan smoothing, Histogram AES (enc,dec)
5	Image processing Signal processing Security Data integrity	Susan edges DTW AES (enc,dec) SHA, CRC32

6.5.2 Results

6.5.2.1 Variation-Aware VA-SNAP Approach and Frequency Scaling

Fig. 6.7a reports the average optimized Millions of Instructions Per Second (MIPS, y-axis) for the range of power budgets (x axis). We plot the results for all chip composition strategy cases shown in the plot labels, with (+FS) and without (no FS) frequency scaling (FS). As can be observed in the plot, for all cases, the MIPS grows proportionally to the budget until the point in which the highest-performing configuration is achieved (i.e., around 1000mW). The *Full NTV* case has a lower IPS performance across all budgets, as the highest-performing NTV configuration (with 16 low-frequency NTV cores) is achievable for small power limits; hence IPS stagnates around 125mW. On the other hand, notice that no *Full STV* configurations are possible with less than 200mW for the required minimum yield of 95%, as they are more power-consuming voltage cases.

As shown by the average results in Fig. 6.7b, the VA-SNAP configurations (composed of a blend of STV and NTV cores), combined with frequency scaling, outperform all other cases due to the availability of more core options that allows exploring power margins with finer granularity. On average, VA-SNAP improves performance over the *Full STV + FS* in 12% and around $3.4\times$ over the *Full NTV+FS* case (geometric mean of gains across all power budgets).

Secondly, post-design frequency scaling improves performance for all chip design approaches. For various minimum yield requirements, the average improvements are shown in Tab. 6.3. More demanding yield levels require smaller chip designs if no FS is adopted. On the other hand, by applying FS, the variation margin assumptions can

Figure 6.7 – MIPS comparison for the different MPSoC composition strategies.

(a) Best average attainable MIPS per budget. (b) Average across all power budgets.

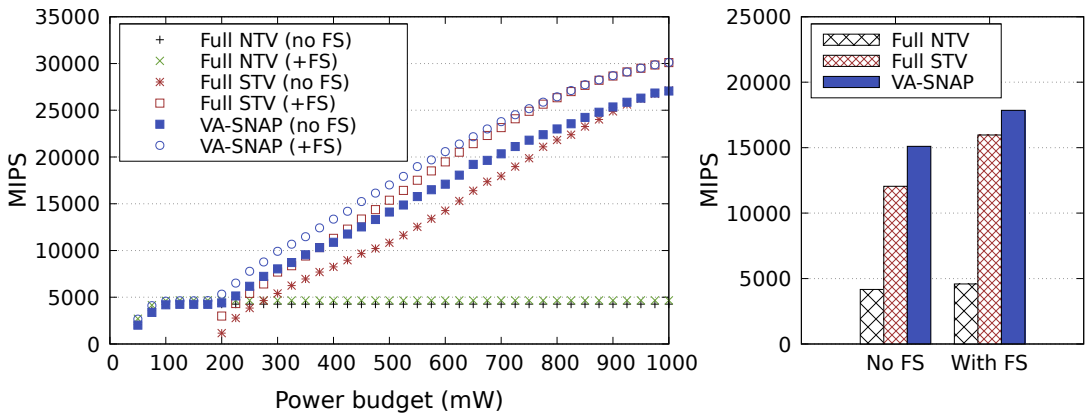


Table 6.3 – Best average achievable MIPS for varying yield requirements (average across all power budgets). Gains are the geometric mean of all gains, across all budgets, provided by frequency scaling.

		Min yield 85%	**Min yield 90%**	**Min yield 95%**	**Min yield 99%**
Full STV	No FS	13470	13377	12049	11232
	With FS	16392	16763	15968	16041
	Gain	21.7%	25.3%	32.5%	42.8%
Full NTV	No FS	4043	4015	3974	3853
	With FS	4431	4431	4427	4249
	Gain	11.7%	13.7%	16.5%	19.4%
VA-SNAP	No FS	16221	15476	15098	14780
	With FS	18091	17759	17849	18459
	Gain	11.5%	14.6%	17.7%	22.4%

be relaxed, allowing for the composition and selection of larger configurations under the power limit. This effect is shown in Tab. 6.3 - as the minimum yield requirement grows, the benefits of the frequency scaling become more pronounced.

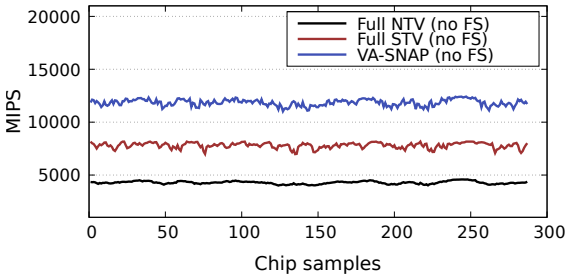
Finally, notice that the frequency scaling is not as effective for *Full NTV* and *VA-SNAP* as it is for *Full STV*. This is due to the presence NTV cores in both cases. Variation is more aggressive at NTV, and variation-afflicted cores with low V_{th} tend to consume more variation-induced *additional* leakage power than the high V_{th} ones save. In other words, under variation, the average static power tends to be higher than the variation-free counterpart. Additionally, the share of the dynamic power of NTV cores is smaller than the static one (both due to low V_{dd} and V_{th} variation). As a result, the post-design FS tends to be more aggressive for the NTV cores when power capping is needed.

6.5.2.2 Case Study

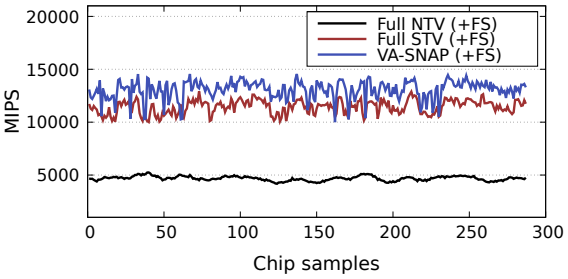
Fig. 6.8 shows an evaluation case optimized for scenario 1 (in Tab. 6.2) for a required power limit of 400mW. For this power limit, we search for the highest-performing configuration for the three chip configuration cases (shown in the plot labels), assuring a minimum yield of 95%. We generate the variation distribution for each configuration and plot the MIPS and power for each sample in Fig 6.8. We evaluate the cases with and without frequency scaling. In Fig. 6.8b, notice that *VA-SNAP (+FS)* consistently provides higher performance than the conventional *Full STV (+FS)* case, with an average MIPS improvement of 16.5% with only 1.6% increase in average power dissipation, measured from the power distribution in Fig. 6.8d.

Figure 6.8 – MIPS and power samples for optimized chips aiming a power limit of 400mW, with and without FS.

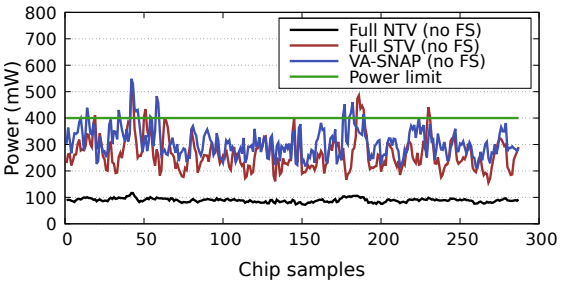
(a) MIPS of chip samples (no FS).



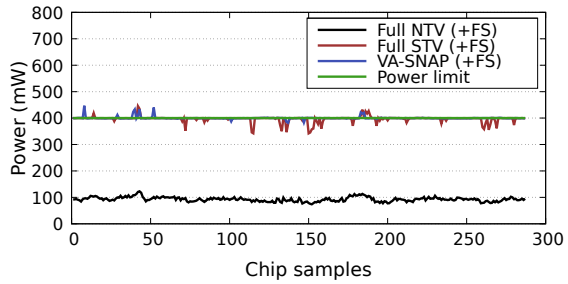
(b) MIPS of chip samples (+FS)



(c) Power of chip samples (no FS).



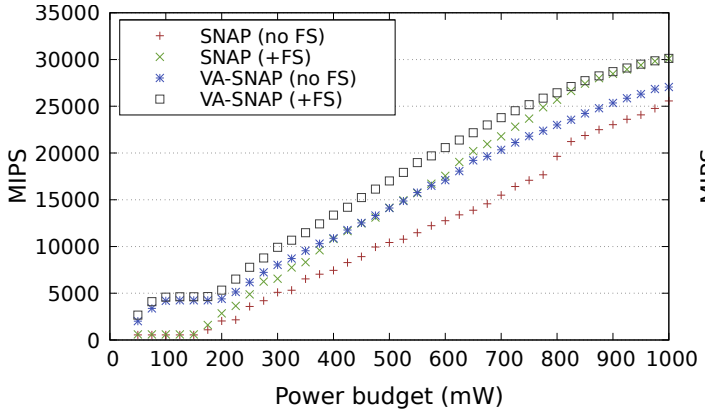
(d) Power of chip samples (+FS).



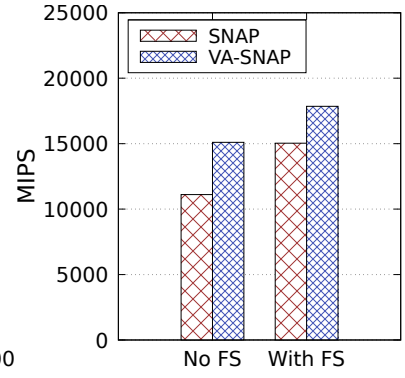
Notice that when frequency scaling is not adopted (Figs. 6.8a and 6.8c), achieving a high minimum yield of 95% requires relying on overly conservative designs, i.e., by adopting low power consuming configurations such that even under the presence of variation most chip samples will not exceed the power limit. This can be seen in the power curves of Fig. 6.8c, which show a high power margin below the power limit. This comes at the cost of performance degradation. Conversely, in Fig. 6.8d, notice that the *Full STV (+FS)* and *VA-SNAP (+FS)* chips have power dissipation very close to the power limit

Figure 6.9 – Average best achievable MIPS for the variation aware (VA-SNAP) and unaware (SNAP) designs, with and without frequency scaling.

(a) Best MIPS for the SNAP and VA-SNAP cases.



(b) Average across all power budgets shown in Fig. 6.9a.



due to optimized frequency scaling calibration. This efficient power margin exploration effectively comes with an associated increase in performance, as shown in Fig. 6.8b.

6.5.2.3 Variation-aware versus Variation-unaware Approaches

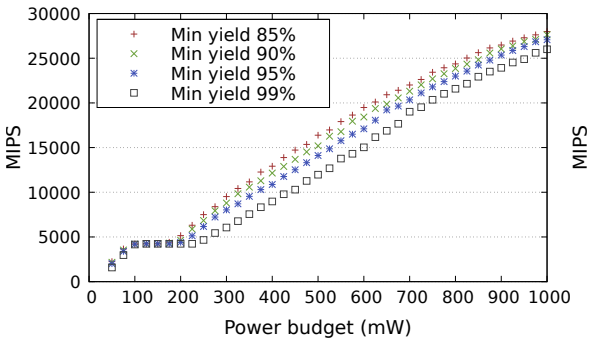
Fig. 6.9 compares the performances of variation-aware (VA-SNAP) and variation-unaware (SNAP) cases with and without FS. On average, if variation is properly harnessed, MIPS improvements of around 51.9% with FS (70.3% without FS) can be achieved compared to the variation-unaware counterparts (geometric mean of gains across all budgets).

6.5.2.4 MIPS for Minimum Yield Requirements

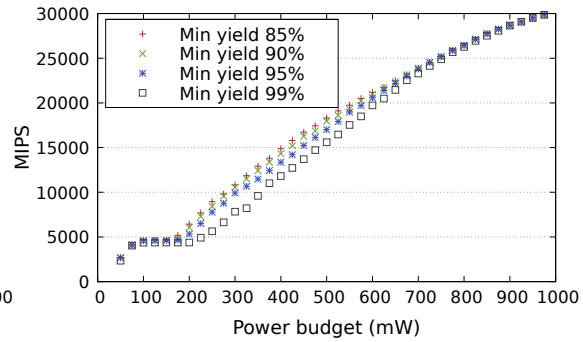
In Fig. 6.10, we report the average best case MIPS for *VA-SNAP* for a set of minimum yield requirements (shown in the plot labels). The critical observation is that as the yield constraint is relaxed, more extensive design choices (with higher power dissipation) are possible for the required budgets, showing a noticeable tradeoff between perfor-

Figure 6.10 – Best attainable MIPS for various yield requirements for the variation-aware VA-SNAP case.

(a) VA-SNAP average best MIPS (no FS)



(b) VA-SNAP average best MIPS (with FS)



(c) Average across all power budgets.

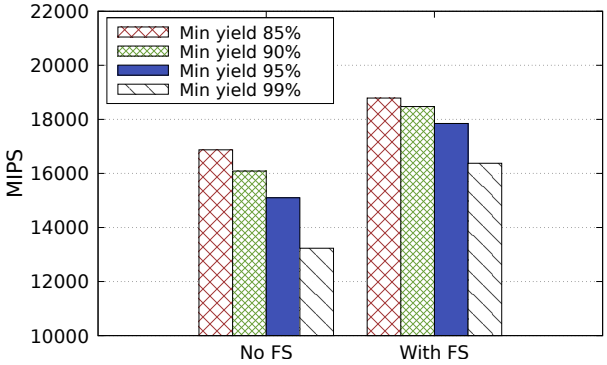
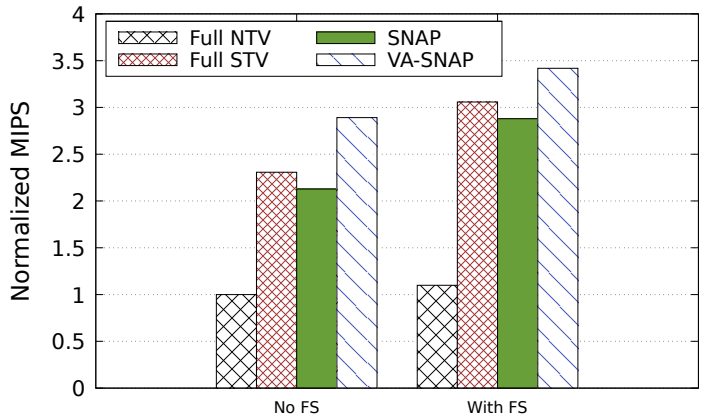


Figure 6.11 – MIPS performance of all evaluated methods, normalized to the Full NTV (No FS) case.



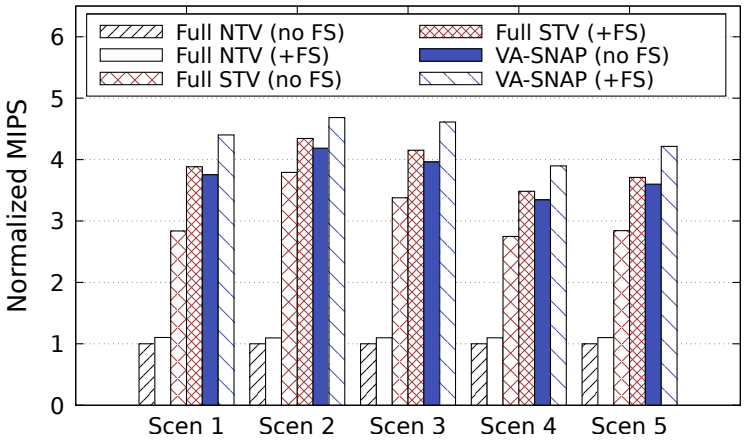
mance/power and yield. For example, the average MIPS increases +27% if the minimum yield is relaxed from 99% to 85% (No FS case) and +15% for the case with FS. Notice that the frequency scaling increases performance for all minimum yield cases, as seen in Fig. 6.10b, since it allows for additional relaxation of power margins during chip design, enabling larger (and with higher performance) configurations for the required yield.

6.5.2.5 Overall Gains Evaluation

The overall optimized (average) MIPS for all evaluated chip design cases is shown in Fig. 6.11, normalized to the *Full NTV (No FS)* (worst) case. Each case renders configurations of maximum performance and a minimum yield of 95%. As it can be observed, performance is progressively improved if a combination of both design-time variation-aware MPSoC composition and careful post-design frequency scaling is adopted.

The results show that a combination of variation-aware designs with a blend of STV and NTV cores (VA-SNAP) has enhanced performance over conventional cases in which architectures operate either entirely at STV or NTV. The variation-aware ILP proposal for the *VA-SNAP* case, combined with frequency scaling, improves performance, on average, around +12% (against *Full STV+FS*), +52% (against *SNAP+FS*), and $3.4\times$ (against *Full NTV+FS*) cases when subject to the same variation constraints and limited to the same power budgets. A minimum yield of 95% is satisfied for the provided configurations in all cases.

Figure 6.12 – Normalized MIPS per each scenario



6.5.2.6 Per-Scenario Results

Fig. 6.12 shows the achievable obtained MIPS (normalized to the *Full NTV (No FS)* case. The gains range from 3.9 up to 4.7 times. Finally, Tab. 6.4 shows the per-scenario MIPS improvements of *SNAP+FS* compared to all explored architectures. In essence, the achievable improvements depend on application-dependent factors such as IPC and mapping solution. Together with the process variation parameters, the power envelope, and yield requirements, such application-dependent aspects influence the efficiency of frequency scaling for the given chips.

Table 6.4 – MIPS gains of VA-SNAP (+FS), per scenario, against all other chip design cases.

	Vs Full NTV (no FS)	**Vs Full NTV (+FS)**	**Vs Full STV (no FS)**	**Vs Full STV (+FS)**	**Vs VA-SNAP (no FS)**
Scenario 1	4.4×	3.9×	+55.1%	+13.3%	+17.3%
Scenario 2	4.7×	4.3×	+23.6%	+7.9%	+11.9%
Scenario 3	4.6×	4.2×	+36.5%	+11.1%	+16.4%
Scenario 4	3.9×	3.5×	+41.8%	+12.0%	+16.5%
Scenario 5	4.3×	3.8×	+48.3%	+13.7%	+17.2%

7 CONCLUSIONS

This work proposes optimization methods to improve key requirements for heterogeneous systems, such as reliability, performance, and energy and efficiency. We also propose a methodology to mitigate process variations constraints on heterogeneous chips aiming to improve performance and energy efficiency while also maintaining yield requirements.

7.1 Addressing Reliability

In the first part of the work, we leverage application and hardware heterogeneity (at the level of core microarchitecture) and propose a mapping strategy that improves the system's resilience to soft errors, measured in terms of Mean Workload To Failure (MWTF). We propose a learning method (by adopting an Artificial Neural Network) that learns application-dependent Architectural Vulnerability Factor (AVF) from core pipeline utilization. We then use the learned AVF outputs to guide the application mapping decisions (during runtime) that provide application-to-core mappings that are very close to the optimal mapping in terms of MWTF.

To evaluate our strategy, we experiment with different configurations of heterogeneous RISC-V cores and compare the achievable MWTF of prediction-based mappings against the optimal oracle. We resume this part of the thesis by highlighting the following conclusions:

1. We propose an ANN-based AVF estimation methodology to infer per-core AVF from application-dependent hardware counters. The ANN runs at software-level and imposes small performance overheads of around 8 K cycles per inference;
2. With the given ANN, we perform MWTF-oriented application mappings to heterogeneous cores. When compared to the oracle mappings, our prediction-based mappings offer MWTF as close as 4.9 percent (max of 6.6 percent) to the oracle solutions on heterogeneous systems;
3. The proposed prediction-based mappings, combined with heterogeneous chip compositions, provide improved MWTF compared to homogeneous systems (average improvement of 14 percent) while also increasing the MWTF/energy tradeoff (average improvement of 6.7 percent).

7.2 Improving Performance and Energy Efficiency with NTV Edge Devices

Aiming at improving performance and energy efficiency under power-constrained scenarios, we provide a methodology (*SNAP*) that performs performance-oriented mappings to heterogeneous and efficient chip designs. The chips are effectively designed by considering selective and efficient use of Near-Threshold Voltage (NTV).

Our approach consists in effectively combining both NTV and STV/conventional cores in the same chip. To achieve very efficient designs under a power limit, we adopt knapsack-like design space exploration to achieve solutions that provide efficient core compositions. By restricting the chip design space to a power limit, we propose an Integer Linear Programming (ILP) approach to maximize the chip-level instruction throughput by exploring heterogeneous cores. The solved solutions provide the most performance- and energy-efficient chips (with an optimized number of each core type and voltage settings) under the required power limit.

We then perform experiments with heterogeneous RISC-V chips and show that designs composed of mixes of NTV and STV cores (*SNAP*) outperform conventional designs that operate fully at STV (or fully at NTV) in terms of chip-level instruction throughput (IPS) and area efficiency (IPS/area). We conclude this part of the thesis as follows:

1. Designs that operate fully at NTV levels have very poor area efficiency due to the low-frequency cores. Our approach mitigates such issue and provides designs with 3.9 times better area efficiency (up to 5.2 times) when compared to standard approaches that operate entirely at NTV;
2. We evaluate the achievable performance in terms of chip-level instruction throughput (IPS). We show that our approach provides +13.3 percent (average) IPS improvements over conventional (full STV) architectures (up to 83 percent). Compared to standard approaches operating fully at NTV, average IPS gains of 3.4 times are achieved (up to 6.3 times).

7.3 Addressing Process Variations with NTV Edge Devices

We augment our previously proposed chip design approach with process variation models and provide a two-step design approach to improve edge-based processors'

performance and energy efficiency restricted to power limits and process variation constraints while maintaining minimum yield requirements (*VA-SNAP*). First, we populate our previous ILP method for chip composition with process variation information. Secondly, we propose an efficient post-design frequency scaling mechanism to either cope with unwanted delay and power variations unseen during fabrication (power capping) or, if possible, to perform variation-enabled frequency boosting to improve performance. Under restricted power limits, the *VA-SNAP* methodology guides towards proper chip configurations that maximize the system's multi-task instruction throughput under process variation scenarios.

We experiment with RISC-V heterogeneous chips by performing performance-oriented mappings to evaluate how *VA-SNAP* performs when compared to standard (*Full STV/NTV*) designs. The following conclusions can be drawn from our experiments:

1. Our frequency scaling approach improves IPS of all explored chip configuration cases (*Full STV/NTV* and *VA-SNAP*). For example, improvements of 32.5 percent (16.5 percent) are achieved for *Full STV* (*Full NTV*) architectures and 17.7 percent for *VA-SNAP*. Secondly, frequency scaling assures minimum yield requirements (95 percent for the previous example);
2. Chip design should effectively consider process variation models both during design- and post-design time. If a minimum required yield must be kept under a variation scenario, our approach can improve performance, on average, by 51.9 percent (if combined with frequency scaling) when compared to variation-agnostic designs;
3. We show that the efficient use of selective NTV, combined with frequency scaling, allows for fine-grain exploration of power slacks under a power limit. When compared to standard *Full STV (Full NTV)* architectures (all using frequency scaling), *VA-SNAP* provides, on average, +12 percent (3.4 times) IPS improvements under the same power limit while also maintaining similar yield levels.

7.4 Future Work

Future proposals could address the limitations of the current work. For example, applications comprise multiple heterogeneous tasks that vary in minimum performance requirements and task-depended reliability criticality. This issue may become a conundrum in a chip design composed of mixed STV/NTV cores with varying system-level raw

soft error rates and increased parametric variations. Thus, a more holistic chip design and mapping approach could be considered according to the following insights:

1. Chip design requirements could rely on multi-objective optimization criteria to balance both performance and reliability, thus providing a more holistic approach to address both SEU reliability and process variations of NTV chips;
2. Secondly, as NTV cores have increased radiation-induced raw error rates, mapping approaches considering chips with both NTV and STV cores could take advantage of such design space to improve reliability. For instance, applications could be classified into varying degrees of criticality, and more robust applications (with a higher degree of fault masking) could be mapped to NTV cores.

Additionally, other proposals could expand the design space by exploring other hardware accelerators available in the Chipyard framework. For example, open-source hardware accelerators for machine learning workloads are already available, such as systolic arrays (GENC et al., 2021). Both SEU reliability and NTV-induced process variations could be explored in the scope of systolic arrays. However, two main issues must be considered.

First, the RTL models of such complex designs tend to suffer from very slow simulation performance, making fault injection campaigns unfeasible for deep machine learning workloads. For that, faster fault injection mechanisms could be proposed at the expense of diminished accuracy.

Secondly, when exploring process variations, it was already shown that variation-induced timing violations on systolic array operations significantly lower neural network accuracy (JIAO et al., 2017). Even under low timing error rates on multiply-accumulate (MAC) operations, there is a significant accuracy drop for both simpler fully connected and more complex deep convolutional networks. Thus, low-cost approaches to tackle timing violations in such designs are an attractive alternative for future works.

7.5 List of Published Papers

The following is a list of published papers over the course of this thesis.

7.5.1 Main Publications

- R. B. Tonetto et al., “A Knapsack Methodology for Hardware-based DMR Protection against Soft Errors in Superscalar Out-of-Order Processors,” 2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC), 2019;
- R. B. Tonetto, H. M. G. de A. Rocha, B. Zatt, A. C. S. Beck and G. L. Nazar, “A Reliability-Oriented Machine Learning Strategy for Heterogeneous Multicore Application Mapping,” 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020;
- R. B. Tonetto, H. M. G. de A. Rocha, G. L. Nazar and A. C. S. Beck, “A Machine Learning Approach for Reliability-Aware Application Mapping for Heterogeneous Multicores,” 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020;
- R. B. Tonetto, A. C. S. Beck and G. L. Nazar, “SNAP: Selective NTV Heterogeneous Architectures for Power-Efficient Edge Computing,” 2022 25th Euromicro Conference on Digital System Design (DSD), Maspalomas, Spain, 2022, pp. 357-364, doi: 10.1109/DSD57027.2022.00055;
- R. B. Tonetto, G. L. Nazar and A. C. S. Beck, “A Variation-Aware Methodology for Improved Processor Designs for the Edge Computing Domain,” , Design Automation for Embedded Systems (DAES), 2023 - (under review);

7.5.2 Publications as a Collaborator

- D. M. Cardoso et al., “Improving Software-based Techniques for Soft Error Mitigation in OoO Superscalar Processors,” 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2019;
- D.M. Cardoso, R. Tonetto, M. Brandalero, G. Nazar, A.C. Beck, J.R. Azambuja. “Exploring the limitations of dataflow SIHFT techniques in out-of-order superscalar processors”, Microelectronics Reliability, Volumes 2019.

REFERENCES

Amid, A. et al. Chipyard: Integrated design, simulation, and implementation framework for custom socs. **IEEE Micro**, v. 40, n. 4, p. 10–21, 2020.

ARM, C. **The Future of Compute, Re-imagined**. Arm corp, 2021. Available from Internet: <https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/arm-dynamiq-technology-for-the-next-era-of-compute>. Accessed in: Apr. 2021.

ASANOVIć, K. et al. **The Rocket Chip Generator**. [S.l.], 2016. Available from Internet: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html>.

AUSTIN, T. M. Diva: a reliable substrate for deep submicron microarchitecture design. In: **MICRO-32'99**. [S.l.: s.n.], 1999. p. 196–207. ISSN 1072-4451.

AZAMBUJA, J. R. et al. Heta: Hybrid error-detection technique using assertions. **IEEE Transactions on Nuclear Science**, v. 60, n. 4, p. 2805–2812, 2013.

Bachrach, J. et al. Chisel: Constructing hardware in a scala embedded language. In: **DAC Design Automation Conference 2012**. [S.l.: s.n.], 2012. p. 1212–1221.

BISWAS, S. K.; MUHURI, P. K.; ROY, U. K. Binary search-based fast scheduling algorithms for reliability-aware energy-efficient task graph scheduling with fault tolerance. **IEEE Transactions on Sustainable Computing**, p. 1–18, 2023.

Bohr, M. A 30 year retrospective on dennard's mosfet scaling paper. **IEEE Solid-State Circuits Society Newsletter**, v. 12, n. 1, p. 11–13, 2007.

Borkar, S. et al. Parameter variations and impact on circuits and microarchitecture. In: **Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)**. [S.l.: s.n.], 2003. p. 338–342.

Bowman, K. A.; Duvall, S. G.; Meindl, J. D. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. **IEEE Journal of Solid-State Circuits**, v. 37, n. 2, p. 183–190, 2002.

BURD, T. et al. A dynamic voltage scaled microprocessor system. **IEEE Journal of Solid-State Circuits**, v. 35, n. 11, p. 1571–1580, 2000.

CARDOSO, D. et al. Exploring the limitations of dataflow sihft techniques in out-of-order superscalar processors. **Microelectronics Reliability**, v. 100-101, p. 113406, 2019. ISSN 0026-2714. 30th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0026271419304767>.

CELIO, C. et al. Broom: An open-source out-of-order processor with resilient low-voltage operation in 28-nm cmos. **IEEE Micro**, v. 39, n. 2, p. 52–60, 2019.

CELIO, C. et al. **BOOM v2: an open-source out-of-order RISC-V core**. [S.l.], 2017. Available from Internet: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-157.html>.

CELIO, C.; PATTERSON, D. A.; ASANOVIć, K. **The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor**. [S.l.], 2015.

Chang, L. et al. Practical strategies for power-efficient computing technologies. **Proceedings of the IEEE**, v. 98, n. 2, p. 215–236, 2010.

CHENG, E. et al. Clear: Cross-layer exploration for architecting resilience - combining hardware and software techniques to tolerate soft errors in processor cores. In: **Proceedings of the 53rd Annual Design Automation Conference**. New York, NY, USA: Association for Computing Machinery, 2016. (DAC '16). ISBN 9781450342360. Available from Internet: <https://doi.org/10.1145/2897937.2897996>.

CHENG, E. et al. Clear: Cross-layer exploration for architecting resilience: Combining hardware and software techniques to tolerate soft errors in processor cores. In: **DAC'16)**. [S.l.: s.n.], 2016. p. 1–6.

DAS, A. et al. Combined dvfs and mapping exploration for lifetime and soft-error susceptibility improvement in mpsocs. In: **2014 Design, Automation & Test in Europe Conference Exhibition (DATE)**. [S.l.: s.n.], 2014. p. 1–6.

Dennard, R. H. et al. Design of ion-implanted mosfet's with very small physical dimensions. **IEEE Journal of Solid-State Circuits**, v. 9, n. 5, p. 256–268, 1974.

Dighe, S. et al. Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. **IEEE Journal of Solid-State Circuits**, v. 46, n. 1, p. 184–193, 2011.

DIXIT, H. D. et al. **Detecting silent data corruptions in the wild**. 2022.

DIXIT, H. D. et al. **Silent Data Corruptions at Scale**. 2021.

Dreslinski, R. G. et al. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. **Proceedings of the IEEE**, v. 98, n. 2, p. 253–266, 2010.

DUQUE, L. A. R.; DIAZ, J. M. M.; YANG, C. Improving mpsoc reliability through adapting runtime task schedule based on time-correlated fault behavior. In: **2015 Design, Automation & Test in Europe Conference Exhibition (DATE)**. [S.l.: s.n.], 2015. p. 818–823.

EDMONDS, J. Paths, trees, and flowers. **Canadian Journal of Mathematics**, Cambridge University Press, v. 17, p. 449–467, 1965.

ENZ, C. C.; KRUMMENACHER, F.; VITTOZ, E. A. An analytical mos transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications. **Analog Integr. Circuits Signal Process.**, Kluwer Academic Publishers, USA, v. 8, n. 1, p. 83–114, jul. 1995. ISSN 0925-1030. Available from Internet: <https://doi.org/10.1007/BF01239381>.

Ernst, D. et al. Razor: circuit-level correction of timing errors for low-power operation. **IEEE Micro**, v. 24, n. 6, p. 10–20, 2004.

Esmaeilzadeh, H. et al. Dark silicon and the end of multicore scaling. In: **2011 38th Annual International Symposium on Computer Architecture (ISCA)**. [S.l.: s.n.], 2011. p. 365–376.

Fu, X.; Li, T.; Fortes, J. A. B. Soft error vulnerability aware process variation mitigation. In: **2009 IEEE 15th International Symposium on High Performance Computer Architecture**. [S.l.: s.n.], 2009. p. 93–104.

GAMMIE, G. et al. A 45nm 3.5g baseband-and-multimedia application processor using adaptive body-bias and ultra-low-power techniques. In: **2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers**. [S.l.: s.n.], 2008. p. 258–611.

GENC, H. et al. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In: **2021 58th ACM/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2021. p. 769–774.

GENC, H. et al. **Gemmini: An Agile Systolic Array Generator Enabling Systematic Evaluations of Deep-Learning Architectures**. 2019.

GEORGE, N. J. et al. Transient fault models and avf estimation revisited. In: **2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)**. [S.l.: s.n.], 2010. p. 477–486.

GOLANBARI, M. S. et al. A cross-layer approach for resiliency and energy efficiency in near threshold computing. In: **2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2016. p. 1–8.

Gopireddy, B. et al. Scalcore: Designing a core for voltage scalability. In: **HPCA'2016**. [S.l.: s.n.], 2016. p. 681–693.

GUTHAUS, M. et al. Mibench: A free, commercially representative embedded benchmark suite. In: **Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)**. [S.l.: s.n.], 2001. p. 3–14.

HAGHBAYAN, M.-H. et al. Dark silicon aware power management for manycore systems under dynamic workloads. In: **2014 IEEE 32nd International Conference on Computer Design (ICCD)**. [S.l.: s.n.], 2014. p. 509–512.

HANDY, J. **How Many Transistors Have Ever Shipped?** 2014. Available from Internet: <https://www.forbes.com/sites/jimhandy/2014/05/26/how-many-transistors-have-ever-shipped/>. Accessed in: Apr. 2021.

HENKEL, J. et al. Reliable on-chip systems in the nano-era: Lessons learnt and future trends. In: **2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2013. p. 1–10.

HENNESSY, J. L.; PATTERSON, D. A. A new golden age for computer architecture. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 62, n. 2, p. 48–60, jan 2019. ISSN 0001-0782. Available from Internet: <https://doi.org/10.1145/3282307>.

HOA, N. T. et al. Deep reinforcement learning for multi-hop offloading in uav-assisted edge computing. **IEEE Transactions on Vehicular Technology**, p. 1–6, 2023.

Horowitz, M. et al. Scaling, power, and the future of cmos. In: **IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest.** [S.l.: s.n.], 2005. p. 7 pp.–15.

JAHINUZZAMAN, S. M.; SHARIFKHANI, M.; SACHDEV, M. An analytical model for soft error critical charge of nanometric srams. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 17, n. 9, p. 1187–1195, 2009.

JIAO, X. et al. An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations. In: **2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2017. p. 945–950.

KANG, Y. et al. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. **SIGARCH Comput. Archit. News**, Association for Computing Machinery, New York, NY, USA, v. 45, n. 1, p. 615–629, apr 2017. ISSN 0163-5964. Available from Internet: <https://doi.org/10.1145/3093337.3037698>.

KARPUZCU, U. R.; KIM, N. S.; TORRELLAS, J. Coping with parametric variation at near-threshold voltages. **IEEE Micro**, v. 33, n. 4, p. 6–14, 2013.

Karpuzcu, U. R. et al. Varius-ntv: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages. In: **DSN'2012**. [S.l.: s.n.], 2012. p. 1–11.

Karpuzcu, U. R. et al. Energysmart: Toward energy-efficient manycores for near-threshold computing. In: **2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)**. [S.l.: s.n.], 2013. p. 542–553.

Kaul, H. et al. Near-threshold voltage (ntv) design — opportunities and challenges. In: **DAC Design Automation Conference 2012**. [S.l.: s.n.], 2012. p. 1149–1154.

Khdr, H. et al. Power density-aware resource management for heterogeneous tiled multicores. **IEEE Transactions on Computers**, v. 66, n. 3, p. 488–501, March 2017. ISSN 2326-3814.

Kriebel, F. et al. Aser: Adaptive soft error resilience for reliability-heterogeneous processors in the dark silicon era. In: **2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2014. p. 1–6.

KUMAR, R. et al. Single-isa heterogeneous multi-core architectures: the potential for processor power reduction. In: **Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.** [S.l.: s.n.], 2003. p. 81–92.

KUMAR, R. et al. Heterogeneous chip multiprocessors. **Computer**, v. 38, n. 11, p. 32–38, 2005.

KUMAR, R. et al. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In: **Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.** [S.l.: s.n.], 2004. p. 64–75.

Lee, Y. et al. A 45nm 1.3ghz 16.7 double-precision gflops/w risc-v processor with vector accelerators. In: **ESSCIRC 2014 - 40th European Solid State Circuits Conference (ESSCIRC)**. [S.l.: s.n.], 2014. p. 199–202.

LEVEUGLE, R. et al. Statistical fault injection: Quantified error and confidence. In: **2009 Design, Automation & Test in Europe Conference Exhibition**. [S.l.: s.n.], 2009. p. 502–506.

LI, E.; ZHOU, Z.; CHEN, X. Edge intelligence: On-demand deep learning model co-inference with device-edge synergy. In: **Proceedings of the 2018 Workshop on Mobile Edge Communications**. New York, NY, USA: Association for Computing Machinery, 2018. (MECOMM'18), p. 31–36. ISBN 9781450359061. Available from Internet: <https://doi.org/10.1145/3229556.3229562>.

LINDOSO, A. et al. A hybrid fault-tolerant leon3 soft core processor implemented in low-end sram fpga. **TNS'17**, v. 64, n. 1, p. 374–381, Jan 2017. ISSN 0018-9499.

MAITI, S.; KAPADIA, N.; PASRICHA, S. Process variation aware dynamic power management in multicore systems with extended range voltage/frequency scaling. In: **2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)**. [S.l.: s.n.], 2015. p. 1–4.

Markovic, D. et al. Ultralow-power design in near-threshold region. **Proceedings of the IEEE**, v. 98, n. 2, p. 237–252, 2010.

MARTINS, M. et al. Open cell library in 15nm freepdk technology. In: **Proceedings of the 2015 Symposium on International Symposium on Physical Design**. New York, NY, USA: Association for Computing Machinery, 2015. (ISPD '15), p. 171–178. ISBN 9781450333993. Available from Internet: <https://doi.org/10.1145/2717764.2717783>.

MARTíNEZ-ÁLVAREZ, A. et al. A hardware-software approach for on-line soft error mitigation in interrupt-driven applications. **IEEE Transactions on Dependable and Secure Computing**, v. 13, n. 4, p. 502–508, 2016.

MITRA, S. et al. The resilience wall: Cross-layer solution strategies. In: **Proceedings of Technical Program - 2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)**. [S.l.: s.n.], 2014. p. 1–11.

MITTAL, S. A survey of techniques for approximate computing. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 48, n. 4, mar. 2016. ISSN 0360-0300. Available from Internet: <https://doi.org/10.1145/2893356>.

MUKHERJEE, S. et al. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In: **Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.** [S.l.: s.n.], 2003. p. 29–40.

NAITHANI, A. et al. Reliability-aware scheduling on heterogeneous multicore processors. In: **2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)**. [S.l.: s.n.], 2017. p. 397–408.

OH, N.; SHIRVANI, P. P.; MCCLUSKEY, E. J. Error detection by duplicated instructions in super-scalar processors. **IEEE Trans. on Reliability**, v. 51, n. 1, p. 63–75, Mar 2002. ISSN 0018-9529.

OZ, I.; ARSLAN, S. A survey on multithreading alternatives for soft error fault tolerance. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 2, mar 2019. ISSN 0360-0300. Available from Internet: <https://doi.org/10.1145/3302255>.

PEDRAM, M. Power minimization in ic design: Principles and applications. **ACM Trans. Des. Autom. Electron. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 1, n. 1, p. 3–56, jan. 1996. ISSN 1084-4309. Available from Internet: <https://doi.org/10.1145/225871.225877>.

PINCKNEY, N. et al. Assessing the performance limits of parallelized near-threshold computing. In: **DAC Design Automation Conference 2012**. [S.l.: s.n.], 2012. p. 1143–1148.

POUCHET, L.-N.; YUKI, T. **PolyBench/C**. 2011. Available from Internet: <https://sourceforge.net/projects/polybench/>.

RABAEY, J. M. **Digital Integrated Circuits: A Design Perspective**. [S.l.]: Pearson, 1995.

RAGHUNATHAN, B. et al. Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors. In: **2013 Design, Automation & Test in Europe Conference Exhibition (DATE)**. [S.l.: s.n.], 2013. p. 39–44.

RAHMANI, A. M. et al. Reliability-aware runtime power management for many-core systems in the dark silicon era. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 25, n. 2, p. 427–440, 2017.

RATHORE, V. et al. Life guard: A reinforcement learning-based task mapping strategy for performance-centric aging management. In: **2019 56th ACM/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2019. p. 1–6.

REHMAN, S. et al. dtune: Leveraging reliable code generation for adaptive dependability tuning under process variation and aging-induced effects. In: **2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2014. p. 1–6.

REIS, G. et al. Design and evaluation of hybrid fault-detection systems. In: **32nd International Symposium on Computer Architecture (ISCA'05)**. [S.l.: s.n.], 2005. p. 148–159.

RODRIGUES, C. et al. Towards a heterogeneous fault-tolerance architecture based on arm and risc-v processors. In: **IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society**. [S.l.: s.n.], 2019. v. 1, p. 3112–3117.

ROSSI, D. et al. A self-aware architecture for pvt compensation and power nap in near threshold processors. **IEEE Design & Test**, v. 34, n. 6, p. 46–53, 2017.

ROZO, L. et al. Reliability-aware runtime adaption through a statically generated task schedule. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 26, n. 1, p. 11–22, 2018.

RUPP, K. **48 Years of Microprocessor Trend Data**. 2021. Available from Internet: <https://github.com/karlrupp/microprocessor-trend-data>. Accessed in: Apr. 2021.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594.

Salehi, M. et al. Drvs: Power-efficient reliability management through dynamic redundancy and voltage scaling under variations. In: **ISLPED'15**. [S.l.: s.n.], 2015. p. 225–230.

SARTOR, A. L. et al. Exploiting idle hardware to provide low overhead fault tolerance for vliw processors. **J. Emerg. Technol. Comput. Syst.**, ACM, New York, NY, USA, v. 13, n. 2, jan. 2017.

SEEPERS, R. M.; STRYDIS, C.; GAYDADJIEV, G. N. Architecture-level fault-tolerance for biomedical implants. In: **SAMOS'12**. [S.l.: s.n.], 2012.

SEMERARO, G. et al. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In: **Proceedings Eighth International Symposium on High Performance Computer Architecture**. [S.l.: s.n.], 2002. p. 29–40.

Shafaei, A. et al. Fincacti: Architectural analysis and modeling of caches with deeply-scaled finfet devices. In: **2014 IEEE Computer Society Annual Symposium on VLSI**. [S.l.: s.n.], 2014. p. 290–295.

SHAFIQUE, M. et al. Variability-aware dark silicon management in on-chip many-core systems. In: **2015 Design, Automation & Test in Europe Conference Exhibition (DATE)**. [S.l.: s.n.], 2015. p. 387–392.

Shi, W. et al. Edge computing: Vision and challenges. **IEEE Internet of Things Journal**, v. 3, n. 5, p. 637–646, 2016.

SMOLENS, J. C. et al. Efficient resource sharing in concurrent error detecting superscalar microarchitectures. In: **MICRO-37'04**. [S.l.: s.n.], 2004. p. 257–268. ISSN 1072-4451.

STAMELAKOS, I. et al. Variation-aware voltage island formation for power efficient near-threshold manycore architectures. In: **2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)**. [S.l.: s.n.], 2014. p. 304–310.

STAMELAKOS, I. et al. Workload- and process-variation aware voltage/frequency tuning for energy efficient performance sustainability of ntc manycores. **Integration**, v. 65, p. 252–262, 2019. ISSN 0167-9260. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0167926016301626>.

STATISTICS, A. **Android and Google Play statistics**. AppBrain Statistics, 2023. Available from Internet: <https://www.appbrain.com/stats>. Accessed in: Jul. 2023.

TAN, C. et al. Locus: Low-power customizable many-core architecture for wearables. In: **2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)**. [S.l.: s.n.], 2016. p. 1–10.

TARSA, S. J. et al. Post-silicon cpu adaptation made practical using machine learning. In: **2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)**. [S.l.: s.n.], 2019. p. 14–26.

Teodorescu, R.; Torrellas, J. Variation-aware application scheduling and power management for chip multiprocessors. In: **2008 International Symposium on Computer Architecture**. [S.l.: s.n.], 2008. p. 363–374.

TONETTO, R. B.; BECK, A. C. S.; NAZAR, G. L. Snap: Selective ntv heterogeneous architectures for power-efficient edge computing. In: **2022 25th Euromicro Conference on Digital System Design (DSD)**. [S.l.: s.n.], 2022. p. 357–364.

TONETTO, R. B. et al. A knapsack methodology for hardware-based dmr protection against soft errors in superscalar out-of-order processors. In: **2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)**. [S.l.: s.n.], 2019. p. 287–292.

TONETTO, R. B. et al. A machine learning approach for reliability-aware application mapping for heterogeneous multicores. In: **2020 57th ACM/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2020. p. 1–6.

Triggs, Robert. **Apple Bionic SoC**. 2021. Available from Internet: <https://www.androidauthority.com/apple-iphone-a14-bionic-benchmark-1173400/>. Accessed in: Mar. 2022.

VADLAMANI, R. et al. Multicore soft error rate stabilization using adaptive dual modular redundancy. In: **2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)**. [S.l.: s.n.], 2010. p. 27–32.

WALCOTT, K. R.; HUMPHREYS, G.; GURUMURTHI, S. Dynamic prediction of architectural vulnerability from microarchitectural state. **SIGARCH Comput. Archit. News**, Association for Computing Machinery, New York, NY, USA, v. 35, n. 2, p. 516–527, jun 2007. ISSN 0163-5964. Available from Internet: <https://doi.org/10.1145/1273440.1250726>.

WAN, Z. et al. Berry: Bit error robustness for energy-efficient reinforcement learning-based autonomous systems. **arXiv preprint arXiv:2307.10041**, 2023.

WANG, J. et al. On the implication of ntc versus dark silicon on emerging scale-out workloads: The multi-core architecture perspective. **IEEE Transactions on Parallel and Distributed Systems**, v. 28, n. 8, p. 2314–2327, 2017.

WANG, N. et al. Characterizing the effects of transient faults on a high-performance processor pipeline. In: **International Conference on Dependable Systems and Networks, 2004**. [S.l.: s.n.], 2004. p. 61–70.

WANG, N. J.; MAHESRI, A.; PATEL, S. J. Examining ace analysis reliability estimates using fault-injection. In: **Proceedings of the 34th Annual International Symposium on Computer Architecture**. New York, NY, USA: Association for Computing Machinery, 2007. (ISCA '07), p. 460–469. ISBN 9781595937063. Available from Internet: <https://doi.org/10.1145/1250662.1250719>.

Wang, N. J.; Patel, S. J. Restore: symptom based soft error detection in microprocessors. In: **DSN'05**. [S.l.: s.n.], 2005. p. 30–39. ISSN 1530-0889.

Wang, X. et al. Convergence of edge computing and deep learning: A comprehensive survey. **IEEE Communications Surveys Tutorials**, v. 22, n. 2, p. 869–904, 2020.

WU, B.-C.; CHEN, W.-T.; LIU, T.-T. An error-resilient risc-v microprocessor with a fully integrated dc–dc voltage regulator for near-threshold operation in 28-nm cmos. **IEEE Journal of Solid-State Circuits**, p. 1–11, 2023.

Wu, C.-J. et al. Machine learning at facebook: Understanding inference at the edge. In: **HPCA'2019**. [S.l.: s.n.], 2019. p. 331–344.

Zhai, B. et al. Energy efficient near-threshold chip multi-processing. In: **Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED '07)**. [S.l.: s.n.], 2007. p. 32–37.

ZHAO, J. et al. Sonicboom: The 3rd generation berkeley out-of-order machine. May 2020.

APPENDIX A — IMPLEMENTATION DETAILS AND PER-SCENARIO EVALUATION

A.1 Detailed Configurations of the Explored Cores

The configuration files for the three explored BOOM microarchitectures are shown in Figures A.1 (SmallBoom), A.2 (MediumBoom), and A.3 (LargeBoom). The code snippets provide high-level description of the micro-architectural parameters, written in the Chisel Hardware Construction Language (or HCL). The parameters include the pipeline structure sizes (as shown in Tab. 6.1 of Sec. 6.5.1), the instruction and data cache configurations, as well as the branch prediction structure (e.g., global history size, BTB size and associativity, with the TAGE predictor disabled).

Figure A.1 – SmallBoom configuration file (Chisel/Scala code).

```
class WithSmallBooms extends Config((site, here, up) => {
  case BoomTilesKey => up(BoomTilesKey, site) map { b => b.copy(
    core = b.core.copy(
      fetchWidth = 4,
      useCompressed = true,
      decodeWidth = 1,
      numRobEntries = 32,
      issueParams = Seq(
        IssueParams(issueWidth=1, numEntries=8, iqType=IQT_MEM.litValue, dispatchWidth=1),
        IssueParams(issueWidth=1, numEntries=8, iqType=IQT_INT.litValue, dispatchWidth=1),
        IssueParams(issueWidth=1, numEntries=8, iqType=IQT_FP.litValue , dispatchWidth=1)),
      numIntPhysRegisters = 52,
      numFpPhysRegisters = 48,
      numLdqEntries = 8,
      numStqEntries = 8,
      maxBrCount = 4,
      numFetchBufferEntries = 8,
      ftq = FtqParameters(nEntries=16),
      btb = BoomBTBParameters(btbsa=true, densebtb=false, nSets=64, nWays=2,
                              nRAS=8, tagSz=20, bypassCalls=false, rasCheckForEmpty=false),
      bpdBaseOnly = None,
      gshare = Some(GShareParameters(historyLength=11, numSets=2048)),
      tage = None,
      bpdRandom = None,
      nPerfCounters = 2,
      fpu = Some(freechips.rocketchip.tile.FPUParams(sfmaLatency=4, dfmaLatency=4, divSqrt=true)))
    dcache = Some(DCacheParams(rowBits = site(SystemBusKey).beatBits,
                               nSets=64, nWays=4, nMSHRs=2, nTLBEntries=8)),
    icache = Some(ICacheParams(rowBits = site(SystemBusKey).beatBits, nSets=64, nWays=4,
fetchBytes=2*4))
  )}
  case SystemBusKey => up(SystemBusKey, site).copy(beatBytes = 8)
  case XLen => 64
  case MaxHartIdBits => log2Up(site(BoomTilesKey).size)
})
```

Figure A.2 – Medium configuration file (Chisel/Scala code).

```
class WithMediumBooms extends Config((site, here, up) => {
  case BoomTilesKey => up(BoomTilesKey, site) map { b => b.copy(
    core = b.core.copy(
      fetchWidth = 4,
      useCompressed = true,
      decodeWidth = 2,
      numRobEntries = 64,
      issueParams = Seq(
        IssueParams(issueWidth=1, numEntries=16, iqType=IQT_MEM.litValue, dispatchWidth=2),
        IssueParams(issueWidth=2, numEntries=16, iqType=IQT_INT.litValue, dispatchWidth=2),
        IssueParams(issueWidth=1, numEntries=16, iqType=IQT_FP.litValue , dispatchWidth=2)),
      numIntPhysRegisters = 80,
      numFpPhysRegisters = 64,
      numLdqEntries = 16,
      numStqEntries = 16,
      maxBrCount = 8,
      numFetchBufferEntries = 16,
      ftq = FtqParameters(nEntries=32),
      btb = BoomBTBParameters(btbsa=true, densebtb=false, nSets=64, nWays=2,
                              nRAS=8, tagSz=20, bypassCalls=false, rasCheckForEmpty=false),
      bpdBaseOnly = None,
      gshare = Some(GShareParameters(historyLength=23, numSets=4096)),
      tage = None,
      bpdRandom = None,
      nPerfCounters = 6,
      fpu = Some(freechips.rocketchip.tile.FPUParams(sfmaLatency=4, dfmaLatency=4, divSqrt=true)))
    dcache = Some(DCacheParams(rowBits = site(SystemBusKey).beatBits,
                               nSets=64, nWays=4, nMSHRs=2, nTLBEntries=8)),
    icache = Some(ICacheParams(rowBits = site(SystemBusKey).beatBits, nSets=64, nWays=4,
fetchBytes=2*4))
    )}
  case SystemBusKey => up(SystemBusKey, site).copy(beatBytes = 8)
  case XLen => 64
  case MaxHartIdBits => log2Up(site(BoomTilesKey).size)
})
```

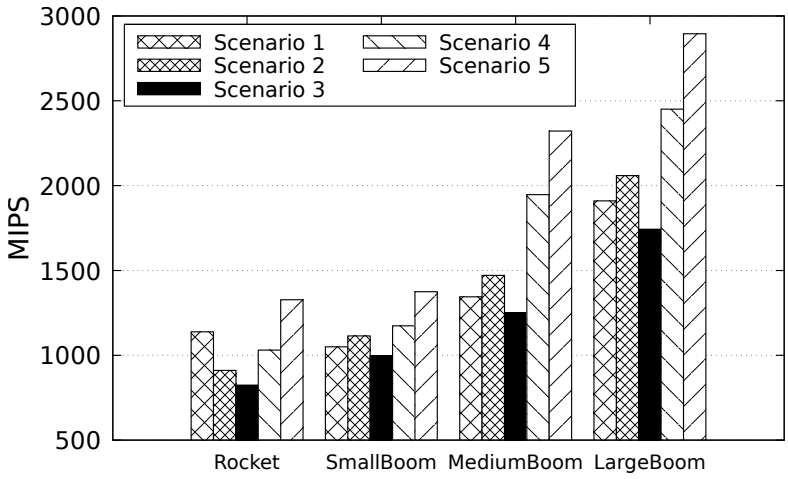
Figure A.3 – LargeBoom configuration file (Chisel/Scala code).

```
class WithLargeBooms extends Config((site, here, up) => {
  case BoomTilesKey => up(BoomTilesKey, site) map { b => b.copy(
    core = b.core.copy(
      fetchWidth = 8,
      useCompressed = true,
      decodeWidth = 3,
      numRobEntries = 96,
      issueParams = Seq(
        IssueParams(issueWidth=1, numEntries=24, iqType=IQT_MEM.litValue, dispatchWidth=3),
        IssueParams(issueWidth=2, numEntries=24, iqType=IQT_INT.litValue, dispatchWidth=3),
        IssueParams(issueWidth=1, numEntries=24, iqType=IQT_FP.litValue , dispatchWidth=3)),
      numIntPhysRegisters = 100,
      numFpPhysRegisters = 96,
      numLdqEntries = 24,
      numStqEntries = 24,
      maxBrCount = 12,
      numFetchBufferEntries = 24,
      ftq = FtqParameters(nEntries=32),
      btb = BoomBTBParameters(btbsa=true, densebtb=false, nSets=512, nWays=4, nRAS=16, tagSz=20),
      bpdBaseOnly = None,
      gshare = Some(GShareParameters(historyLength=23, numSets=4096)),
      tage = None,
      bpdRandom = None,
      fpu = Some(freechips.rocketchip.tile.FPUParams(sfmaLatency=4, dfmaLatency=4, divSqrt=true)))
    dcache = Some(DCacheParams(rowBits = site(SystemBusKey).beatBytes*8,
                               nSets=64, nWays=8, nMSHRs=4, nTLBEntries=16)),
    icache = Some(ICacheParams(fetchBytes = 4*4, rowBits = site(SystemBusKey).beatBytes*8,
nSets=64, nWays=8))
  )}
  case SystemBusKey => up(SystemBusKey, site).copy(beatBytes = 16)
  case XLen => 64
  case MaxHartIdBits => log2Up(site(BoomTilesKey).size)
})
```

A.2 Per-Scenario VA-SNAP Evaluation

For each core configuration, Fig. A.4 depicts the achievable MIPS for each application scenario explored in Chapter 6, in Tab. 6.2. The per-scenario VA-SNAP evaluation is shown in Figs.A.5 to A.9.

Figure A.4 – Per-core attainable MIPS for each scenario explored in Chapter 6 (Tab. 6.2).
(a) STV cores at nominal conditions



(b) NTV cores at nominal conditions

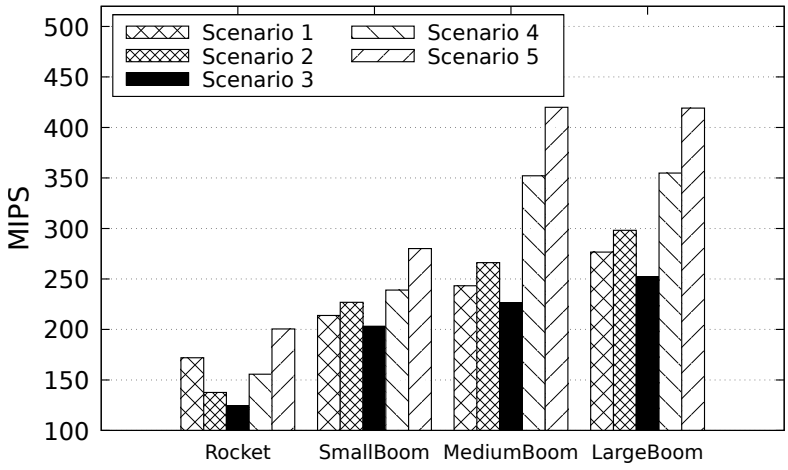


Figure A.5 – Scenario 1.

(a) Best average attainable MIPS per budget. (b) Average across all power budgets.

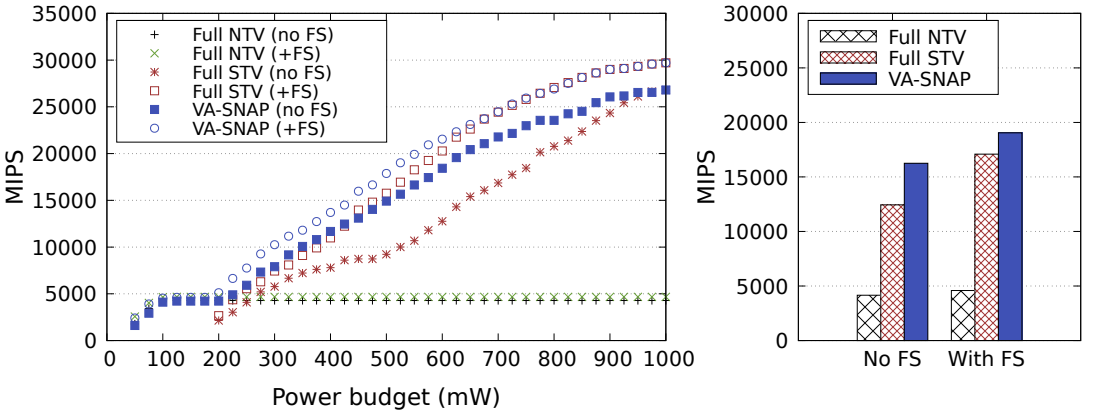


Figure A.6 – Scenario 2.

(a) Best average attainable MIPS per budget. (b) Average across all power budgets.

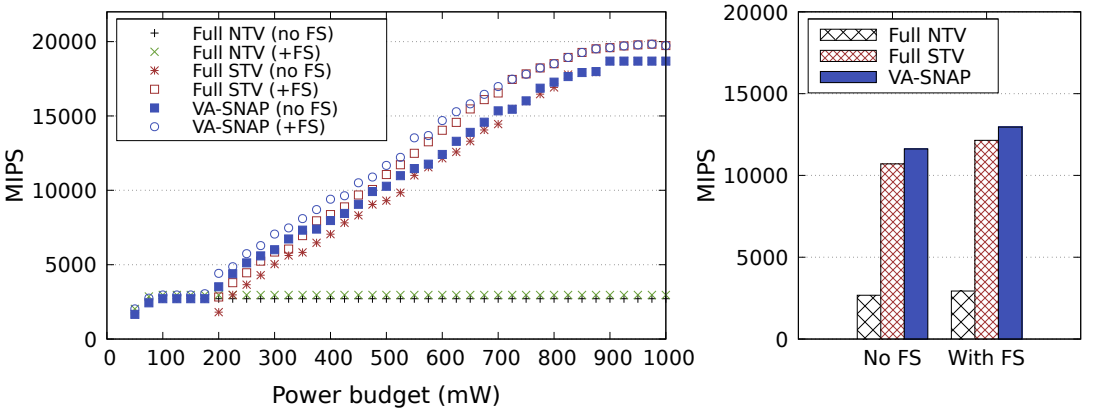


Figure A.7 – Scenario 3.

(a) Best average attainable MIPS per budget. (b) Average across all power budgets.

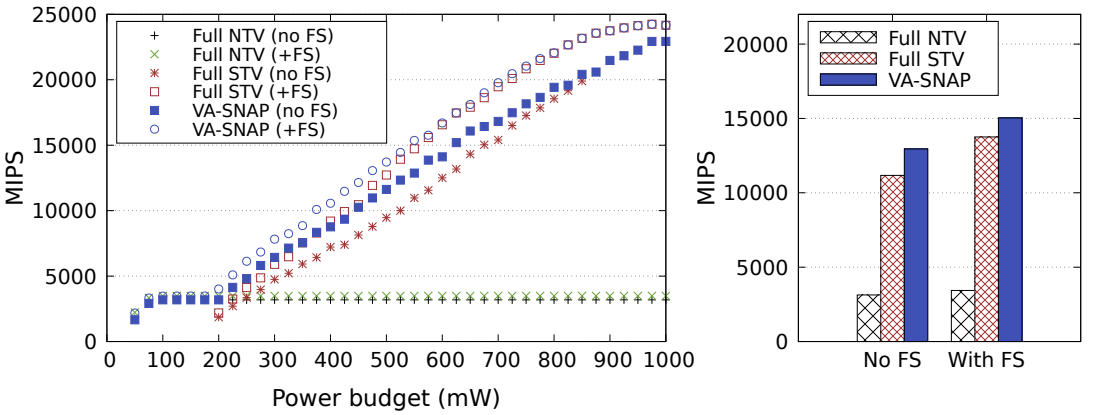


Figure A.8 – Scenario 4.

(a) Best average attainable MIPS per budget. (b) Average across all power budgets.

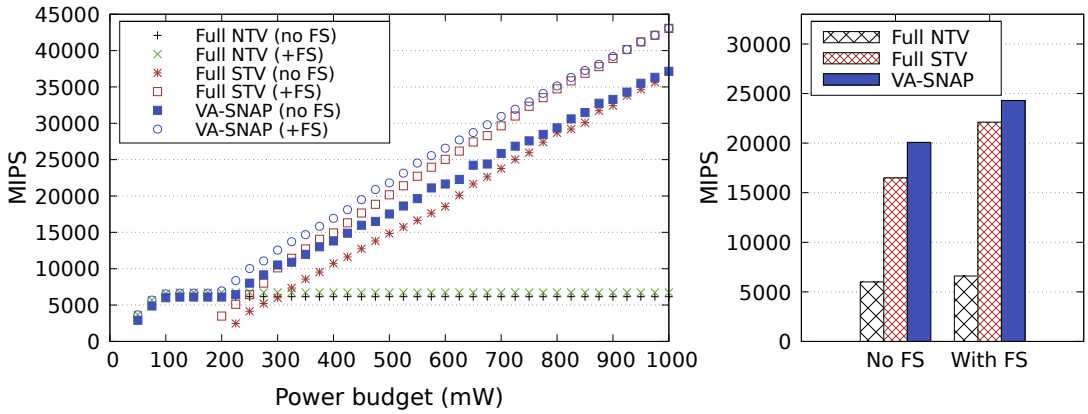
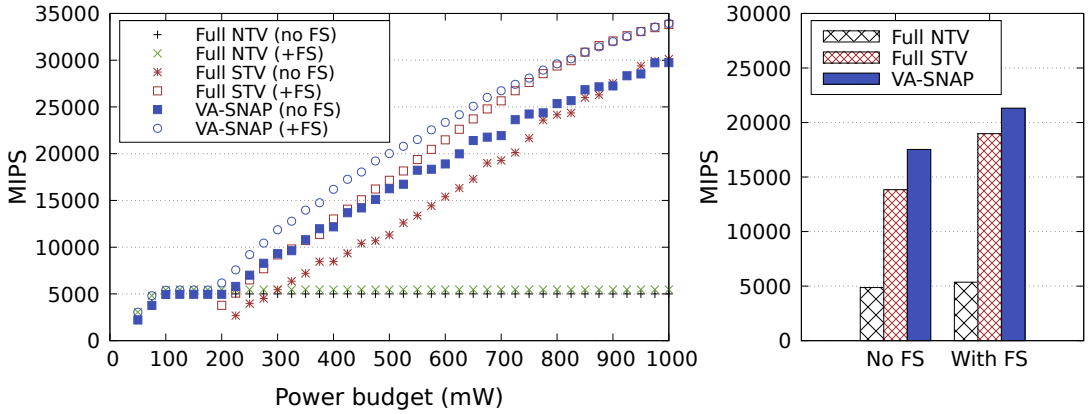


Figure A.9 – Scenario 5.

(a) Best average attainable MIPS per budget. (b) Average across all power budgets.

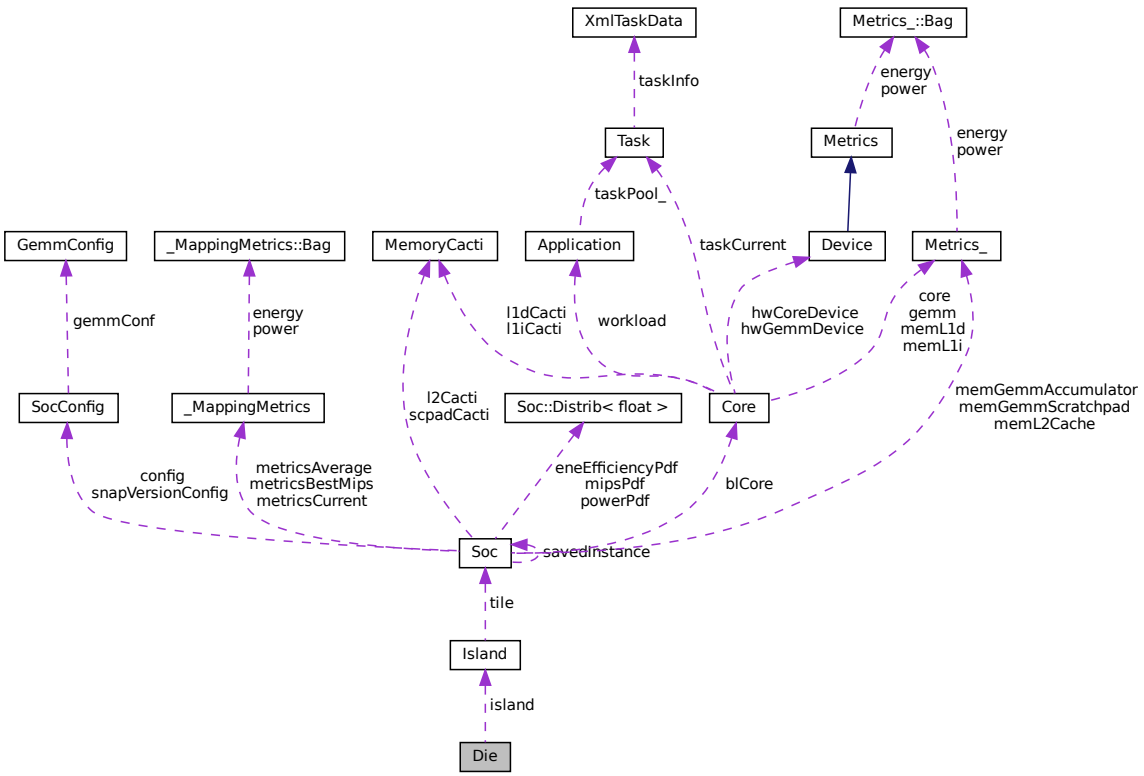


A.3 Class Diagram of the Architecture

Fig. A.10 shows the (simplified) class diagram of the C++ source code developed over the course of this thesis. The diagram describes the structure of a die in the following hierarchical order: $Die \rightarrow Island \rightarrow Soc(tile) \rightarrow Core \rightarrow Device$. Chip (Soc) configurations are defined in the $SocConfig$ class. The cores' properties such as area and power (extracted from the Genus reports) are modeled in the $Device$ class. Each core has also first-level instruction and data cache parameters (class $MemoryCacti$). Applications (class $Application$) are sets of tasks (class $Task$). All tasks and their properties are stored in a XML file (loaded and parsed by the class $XmlTaskData$). The properties include the number of cycles, instructions taken, switching activity, and the number of memory accesses for each memory type when mapped to each core type.

Process variation is simulated by mapping the VARIUS-NTV variation maps to each Die chip instance. The $Soc :: Distrib$ class attribute holds the MIPS and power distribution samples.

Figure A.10 – Class diagram of the source code.

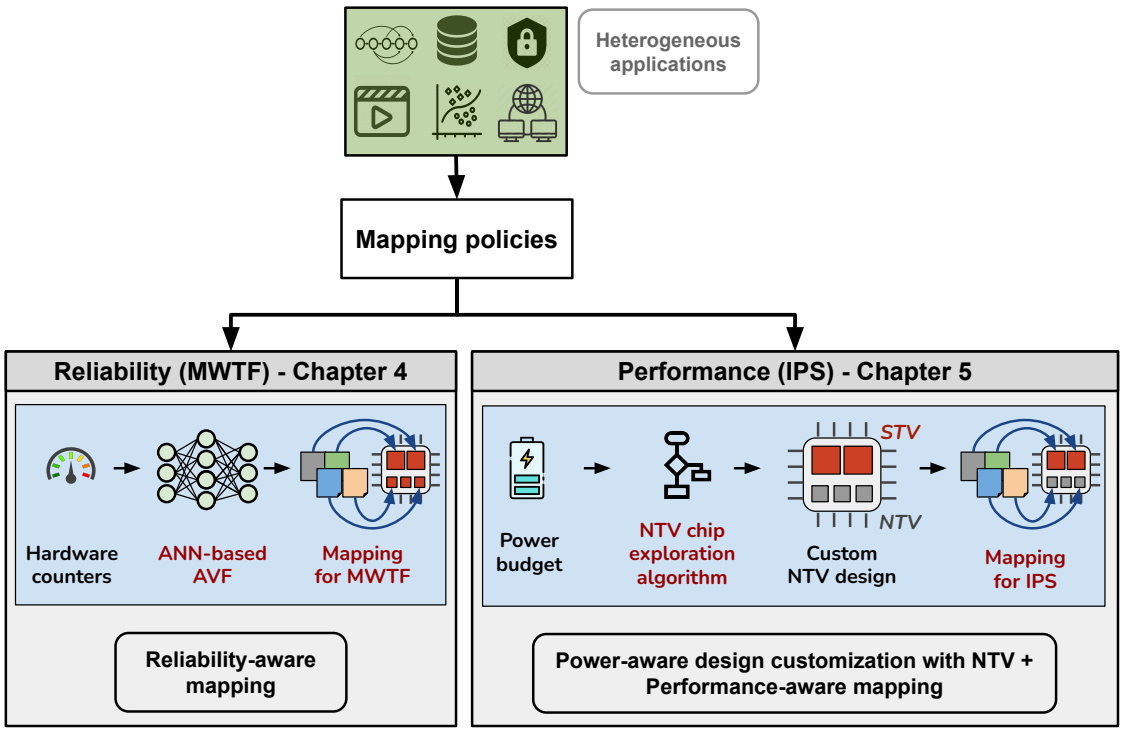


APPENDIX B — UMA METODOLOGIA VISANDO O APRIMORAMENTO DE PROCESSADORES RESTRITOS À VARIAÇÃO DE PROCESSOS E APLICÁVEIS À COMPUTAÇÃO DE BORDA

Este trabalho propõe métodos de otimização para melhorar requisitos-chave de sistemas heterogêneos, como confiabilidade, desempenho, eficiência energética. Também propomos uma metodologia para mitigar as restrições de variações de processo em chips heterogêneos, visando melhorar o desempenho e a eficiência energética, ao mesmo tempo em que se mantêm os requisitos mínimos de yield. Uma visão geral do escopo deste trabalho é mostrada na Fig. B.1.

Figure B.1 – Visão geral desta tese. As contribuições principais estão marcadas em vermelho.

(a) Na primeira parte, exploramos políticas de mapeamento com o objetivo de maximizar a confiabilidade e o desempenho.



(b) A segunda parte tem como objetivo compor designs de chips eficientes em energia e cientes do problema de variações de processo devido ao NTV.

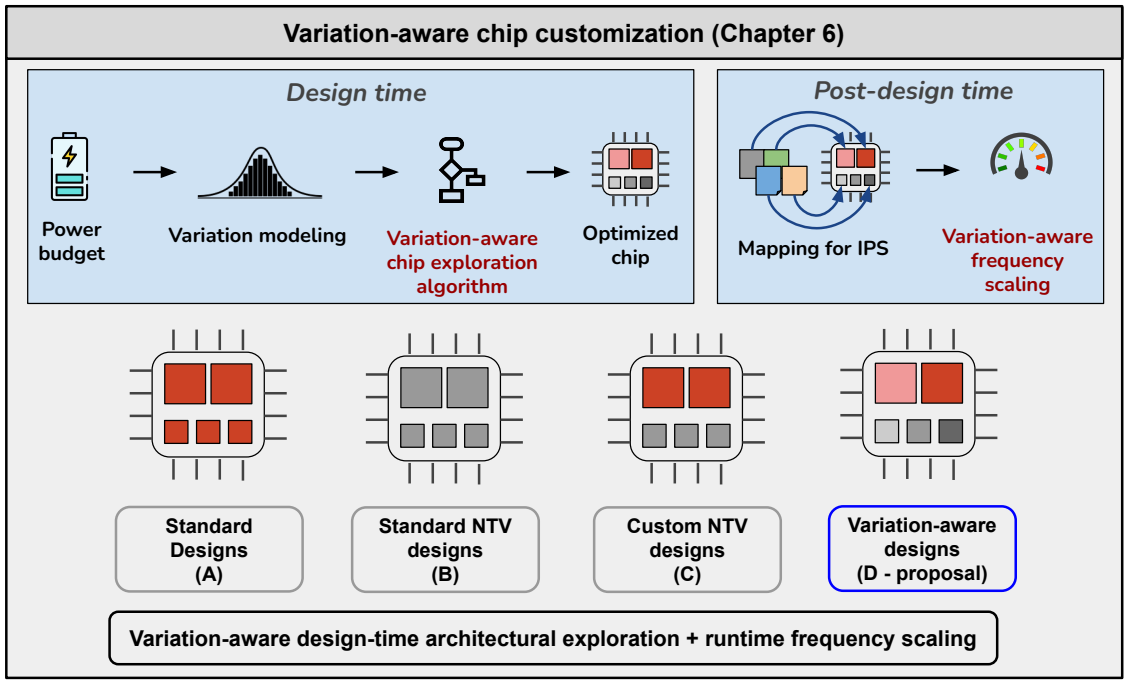
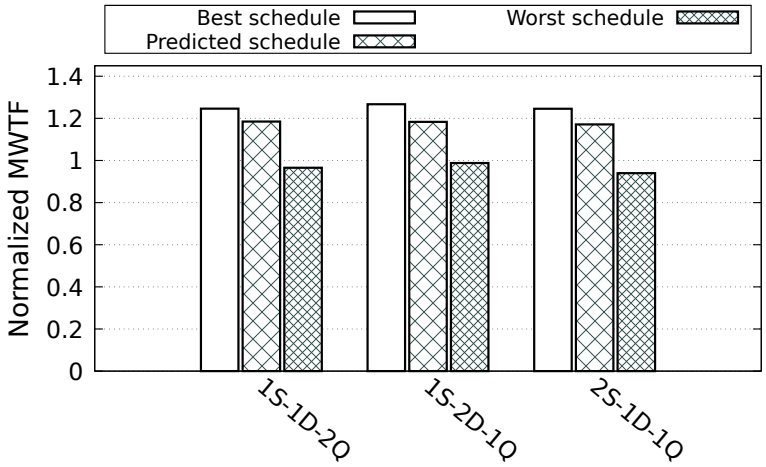


Figure B.2 – Ganhos médios de MWTF para todos os conjuntos de aplicações nos três casos de mapeamento.



B.1 Parte 1: Otimização de Confiabilidade

Na primeira parte do trabalho, aproveitamos a heterogeneidade de aplicativos e hardware (no nível da microarquitetura do núcleo do processador) e propomos uma estratégia de mapeamento que melhora a confiabilidade do sistema em termos de erros transitórios causados por radiação. A métrica considerada é Média de Carga de Trabalho até a Falha (MWTF, do inglês Mean Workload to Failure). Propomos um método de aprendizado (adotando uma Rede Neural Artificial) que aprende o Fator de Vulnerabilidade Arquitetural (AVF) dependente do aplicativo a partir da utilização do pipeline do núcleo. Em seguida, utilizamos as saídas do AVF aprendido para orientar as decisões de mapeamento do aplicativo (durante a execução) que fornecem mapeamentos de aplicativos para núcleos que estão muito próximos do mapeamento ideal em termos de MWTF.

Para avaliar nossa estratégia, experimentamos diferentes configurações de núcleos RISC-V heterogêneos e comparamos a MWTF alcançável dos mapeamentos baseados em previsão com o oráculo ideal. Os principais resultados, em termos de MWTF, são mostrados na Fig. B.2. Em resumo, nesta parte do trabalho, mostramo que:

1. Propomos uma metodologia de estimativa de AVF baseada em ANN para inferir o AVF por núcleo a partir de contadores de hardware dependentes do aplicativo. A ANN é executada em nível de software e impõe pequenos custos de desempenho de cerca de 8 mil ciclos por inferência.
2. Com a ANN treinada, realizamos mapeamentos de aplicativos orientados à MWTF em núcleos heterogêneos. Em comparação com os mapeamentos do oráculo, nossos mapeamentos baseados em previsão oferecem MWTF tão próxima quanto 4,9 por

cento (máximo de 6,6 por cento) em relação às soluções do oráculo em sistemas heterogêneos.

3. Os mapeamentos baseados em previsão propostos, combinados com composições de chips heterogêneos, proporcionam uma melhoria na MWTF em comparação com sistemas homogêneos (melhoria média de 14 por cento), ao mesmo tempo em que aumentam a relação MWTF/energia (melhoria média de 6,7 por cento).

B.2 Parte 2: Otimização de Desempenho e Eficiência Energética

Visando melhorar o desempenho e a eficiência energética em cenários de restrição de energia, fornecemos uma metodologia (*SNAP*) que realiza mapeamentos orientados ao desempenho em designs de chips heterogêneos e eficientes. Os chips são projetados considerando o uso seletivo e eficiente da Tensão Próxima do Limiar (NTV, do inglês Near-Threshold Voltage).

Nossa abordagem, mostrada na Fig. B.3, consiste em combinar efetivamente tanto os núcleos NTV quanto os núcleos STV/conventionais no mesmo chip. Para obter designs muito eficientes sob um limite de potência, adotamos uma exploração do espaço de design semelhante a uma mochila para alcançar soluções que proporcionem composições eficientes de núcleos. Ao restringir o espaço de design do chip a um limite de potência,

Figure B.3 – Visão de nível de arquitetura do chip.

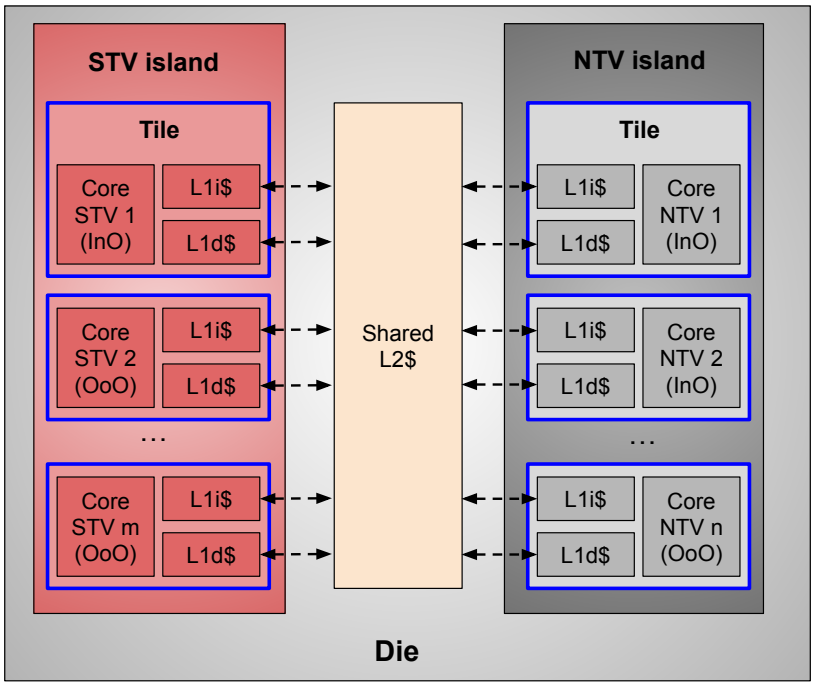
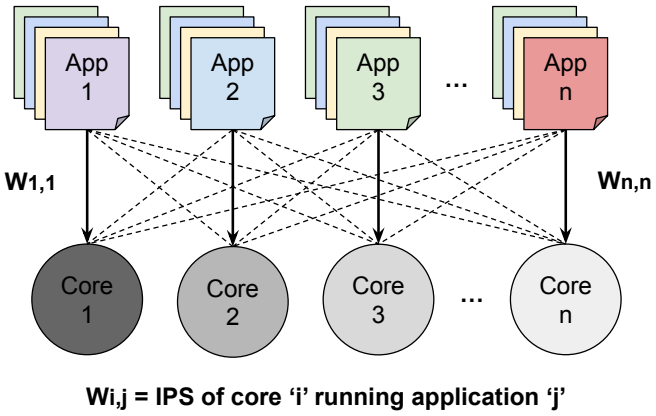


Figure B.4 – O mapeamento de aplicações visa maximizar o MIPS total (milhões de instruções por segundo).



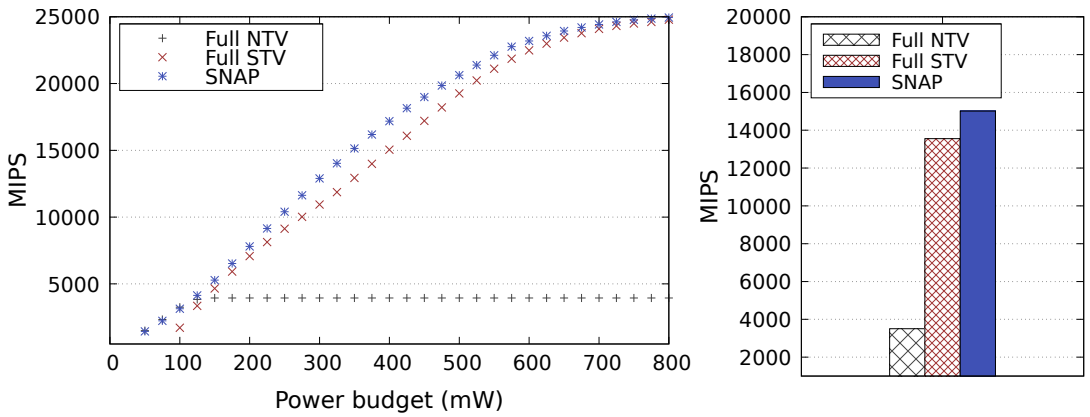
propomos uma abordagem de Programação Linear Inteira (ILP, do inglês Integer Linear Programming) para maximizar o rendimento de instruções em nível de chip, explorando núcleos heterogêneos. As soluções encontradas fornecem os chips mais eficientes em termos de desempenho e energia (com número otimizado de cada tipo de núcleo e configurações de tensão) sob o limite de potência requerido.

Em seguida, realizamos experimentos com chips heterogêneos RISC-V utilizando mapeamentos visando aumento de desempenho, conforme mostrado na Fig. B.4, e mostramos que os designs compostos por combinações de núcleos NTV e STV (*SNAP*) superam os designs convencionais que operam completamente em STV (ou completamente em NTV) em termos de rendimento de instruções em nível de chip (IPS) e eficiência de área (IPS/área). Conforme os resultados mostrados na Fig. B.5, concluímos esta parte da tese da seguinte maneira:

Figure B.5 – Comparação de MIPS para as diferentes estratégias de composição do MPSoC.

(a) Melhor MIPS médio alcançável por limite de potência.

(b) Média entre todos os limites de potência.



1. Os designs que operam totalmente em níveis NTV apresentam uma eficiência de área muito baixa devido aos núcleos de baixa frequência. Nossa abordagem mitiga esse problema e fornece designs com uma eficiência de área 3,9 vezes melhor (até 5,2 vezes) em comparação com abordagens convencionais que operam totalmente em NTV.
2. Avaliamos o desempenho alcançável em termos de rendimento de instruções em nível de chip (IPS). Mostramos que nossa abordagem proporciona melhorias de IPS de +13,3 por cento em média em relação às arquiteturas convencionais (STV completo) (até 83 por cento). Em comparação com abordagens padrão que operam totalmente em NTV, são obtidos ganhos médios de IPS de 3,4 vezes (até 6,3 vezes).

B.3 Parte 3: Abordando Variabilidade de Processos

Aprimoramos nossa abordagem de design de chips proposta anteriormente com modelos de variação de processo e apresentamos uma abordagem de projeto de processadores em duas etapas para melhorar o desempenho e a eficiência energética aplicáveis

Figure B.6 – Camadas do sistema exploradas nesta tese.

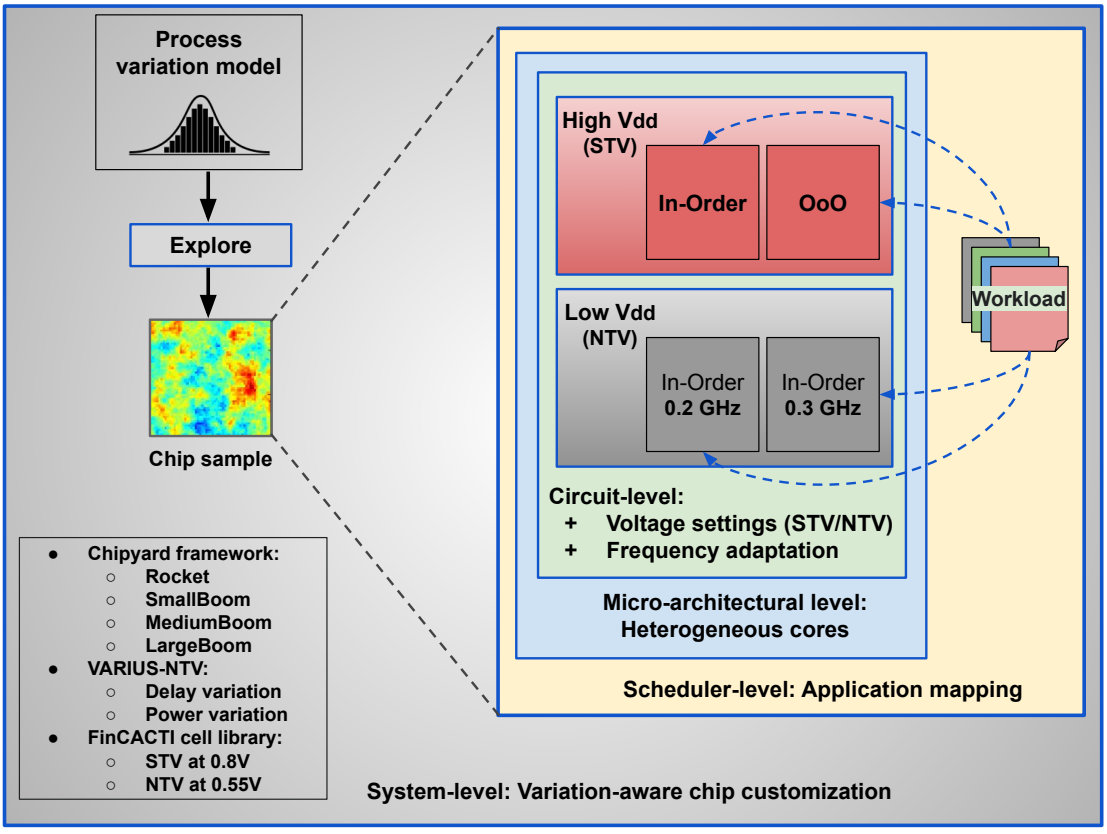
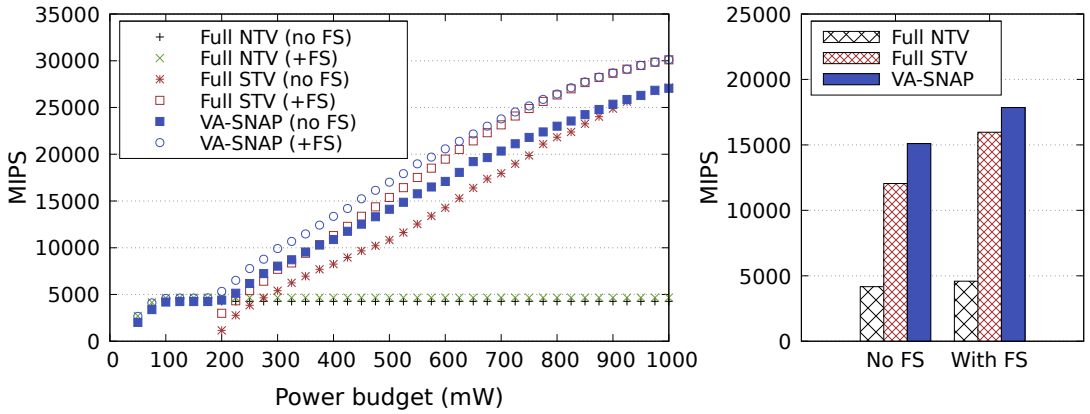


Figure B.7 – Comparação de MIPS para as diferentes estratégias de composição de MPSoC.

(a) Melhor MIPS médio alcançável por limite de potência.

(b) Média entre todos os limites de potência na Fig. B.7a.



ao domínio de borda, restritos a limites de potência e restrições de variação de processo, ao mesmo tempo em que mantemos os requisitos mínimos de rendimento.

A abordagem proposta (*VA-SNAP*) opera em várias camadas do sistema, conforme mostra a Fig. B.6. Primeiro, incorporamos informações de variação de processo ao nosso método ILP anterior para composição de chips. Em segundo lugar, propomos um mecanismo eficiente de ajuste de frequência pós-design para lidar com variações indesejadas de atraso e potência que não são vistas durante a fabricação (limite de potência) ou, se possível, realizar aumento de frequência habilitado para variação para melhorar o desempenho. Sob limites de potência restritos, a metodologia *VA-SNAP* orienta em direção a configurações adequadas de chips que maximizam o rendimento de instruções multitarefa do sistema em cenários de variação de processo.

Realizamos experimentos com chips heterogêneos RISC-V, aplicando mapeamentos orientados ao desempenho para avaliar como o *VA-SNAP* se compara a designs padrão (*Full STV/NTV*). Os principais resultados são mostrados na Fig. B.7. As seguintes conclusões podem ser tiradas de nossos experimentos:

1. Nossa abordagem de ajuste de frequência melhora o rendimento de instruções em todos os casos de configuração de chips explorados (*Full STV/NTV* e *VA-SNAP*). Por exemplo, melhorias de 32,5 por cento (16,5 por cento) são alcançadas para arquiteturas *Full STV* (*Full NTV*), e 17,7 por cento para *VA-SNAP*. Em segundo lugar, o ajuste de frequência assegura os requisitos mínimos de rendimento (95 por cento para o exemplo anterior).
2. O design do chip deve considerar efetivamente modelos de variação de processo tanto durante o tempo de design quanto após o design. Se um rendimento mínimo

exigido deve ser mantido em um cenário de variação, nossa abordagem pode melhorar o desempenho, em média, em 51,9 por cento (se combinada com ajuste de frequência) em comparação com designs que não consideram a variação.

3. Mostramos que o uso eficiente da NTV seletiva, combinado com o ajuste de frequência, permite a exploração granular dos recursos de potência sob um limite de potência. Em comparação com arquiteturas padrão *Full STV (Full NTV)* (todas usando ajuste de frequência), *VA-SNAP* oferece, em média, melhorias de IPS de +12 por cento (3,4 vezes) sob o mesmo limite de potência, mantendo níveis de rendimento semelhantes.