Non-Functional Considerations of Time-Randomized Processor Architectures

David Trilla Rodríguez

Universitat Politècnica de Catalunya

Computer Architecture Department

PhD thesis

Doctoral programme on Computer Architecture

21st of September, 2020

Non-Functional Considerations of Time-Randomized Processor Architectures

David Trilla Rodríguez

September 2020

Universitat Politècnica de Catalunya
Computer Architecture Department

A thesis submitted in fulfillment of
the requirements for the degree of
Doctor of Philosophy in Computer Architecture

Advisor:	Jaume Abella, PhD, Barcelona Supercomputing Center
Co-Advisor:	Carles Hernández, PhD, Barcelona Supercomputing Center
Tutor:	Francisco J. Cazorla, PhD, Universitat Politècnica de Catalunya

To my parents,
for their infinite patience and unconditional support.

Acknowledgements

Now that the Ph.D. endeavor is reaching it's end, I can look back and reflect on what lifted a lot of the weight of this enterprise from my shoulders. I want devote this page to the people that helped me in easing the hardships of reaching the highest step in the educational stair, my Ph.D.

I want to start by thanking my advisors, Francisco Javier, Jaume and Carles for all the opportunities and knowledge they have given me and all the trust they have deposited on me. Also to my CAOS collegues during the journey, Mikel, Javier, María, Leonidas, and many more who came and went through the group. To my university colleagues Pedro, Albert, Cristóbal and Constan, guilty of most of the procrastination but also the moments of relief.

I also want to thank my hard-core infancy friends from my home town, Palau-solità i Plegamans. Sadly decorum impedes me writing down our "gang's" name, but here are yours Oriol, Ivo, both Davids and Hector. Most of the fun in my life involved at least one of them.

I also want to dedicate some words to many friends I met along the way during all sorts of stages of my life. Mariona, Ariane, Gemma, Alvaro, Sandra, Irene, Cristina, Kayla for all the support during my adventure in the U.S.A. and also my sister Olga for all the fun we have had. I am leaving many names behind for brevity, but be assured I hold you all well in my mind.

I am also thankful of my grandparents. My grandmothers Isabel and Maria for their kindness and benevolence, and my grandfathers for their curiosity, I still hold my memories of *yayo* Tomás' crazy inventions with scrap electronics and how *avi* Joan introduced me to informatics and taught me how to build my first personal computer. I am certain you are much at fault for my incline to computer related topics.

And finally, but most importantly, I want to thank my parents Isabel and Juan Luis, who raised, taught and bestowed on me an attitude that allows me to pursue higher standards and overcome failures and hardships. I want to thank them for their sacrifices and giving me everything despite getting much less in return. Thanks to you I can thrive and be content with my life.

Thank you all, this thesis is also for you.
Muchas gracias a todos, esta tesis también es para vosotros.
Moltes gracies a tots, aquesta tesi també és per a vosaltres.

Abstract

Critical Real-Time Embedded Systems (CRTES) are the subset of embedded systems with timing constraints whose miss behavior can endanger human lives or expensive equipment. To provide evidence of correctness, CRTES are designed, implemented and deployed in adherence to safety standards and certification regulations. To that end, CRTES follow strict Validation & Verification (V&V) procedures of their functional and non-functional properties. One of the most important non-functional properties is *timing*, which builds on computing the worst-case execution time of tasks and a schedule of tasks so that the overall system timing behavior is correct. However, the use of more complex hardware and software to satisfy CRTES unprecedented performance requirements, heavily increase the cost of V&V.

For timing V&V, statistical techniques, like Measurement-Based Probabilistic Timing Analysis (MBPTA) help to address the complexity of hardware and software in CRTES. To that end, they benefit from randomization of temporal behavior at the hardware level. In this line, Time-Randomized Processors (TRP) contain timing V&V costs by breaking systematic pathological behaviors and enabling MBPTA applicability.

In the context of TRP, this thesis shows that hardware and software designs incorporating randomization can not only successfully tackle the existing timing analysis problem, but also provide helpful properties to other emerging non-functional metrics key in CRTES like reliability, security and energy. For reliability, we show that TRP are naturally resilient against hardware aging effects and voltage noise and we add up to such resilience by improving its design. Also, TRP hinders security threats and intrusions by breaking and mangling the deterministic association between memory mapping and access time and we develop a framework for secure automotive operation. Finally for energy, we introduce a taxonomy to guide the future challenges for worst-case energy estimation and make the first steps towards the use of MBPTA-like methodology to address worst-case energy estimation under the presence of process variation. Moreover this thesis also shows that together with the application of MBPTA-like methodology, TRP also naturally expose and break pathological energy consumption patterns and help in validating and accounting instantaneous peak power demands. In summary, this thesis pioneers several aspects of the use of TRP to address the emerging challenges that CRTES face in the reliability, security and energy domains.

Contents

Acknowledgements vii

Abstract ix

Contents xi

List of Figures xv

List of Tables xvii

List of Abbreviations xix

I Introduction 1

1 Introduction 3
- 1.1 A Changing Paradigm in CRTES . . . 3
- 1.2 Emerging Non-functional Requirements . . . 5
  - 1.2.1 Software Timing . . . 5
  - 1.2.2 Hardware Reliability . . . 6
  - 1.2.3 Security . . . 7
  - 1.2.4 Energy . . . 7
- 1.3 Summary of Challenges in future CRTES . . . 8
- 1.4 Time-Randomized Processor Architectures . . . 9
- 1.5 Hypothesis . . . 10
- 1.6 Contributions . . . 11
  - 1.6.1 Reliability . . . 12
  - 1.6.2 Security . . . 12
  - 1.6.3 Energy . . . 13
- 1.7 Structure of the Thesis . . . 13
- 1.8 List of Publications . . . 14
  - 1.8.1 Informal Publications . . . 15

2 Background 17
- 2.1 Timing Analysis . . . 17
  - 2.1.1 Deterministic Timing Analysis . . . 18

2.1.2 Probabilistic Timing Analysis 20
2.2 MBPTA Methodology and Requirements 21
2.2.1 Statistical Analysis 22
2.3 Processor and Code Architectures favoring Randomization 23
2.3.1 Software Randomization 24
2.3.2 Time-Randomized Caches 25
2.3.3 Randomized Arbitration Policies 29
2.4 Non-Functional Metrics 29
2.4.1 Reliability 30
2.4.2 Security 31
2.4.3 Energy 32

3 Experimental Setup 35

3.1 Architectures 35
3.2 Tools & Simulators 38
3.3 Benchmarks & Applications 38
3.4 Methodology 39

II Reliability in CRTES 44

4 Enhanced Randomized Cache Designs for Improved Aging and Reliability 45

4.1 Introduction 45
4.2 Enhanced Aging-Friendly Random Cache Placement 46
4.2.1 Random Modulo Set Distribution 46
4.2.2 Randomizing Set Distribution 48
4.2.3 Hash Random Placement Set Distribution 50
4.3 Evaluation 51
4.3.1 Methodology 51
4.3.2 Set distribution 52
4.3.3 Lifetime 55
4.3.4 Delay impact 59
4.4 Resonant Voltage Noise Resilience 59
4.5 Related Work 60
4.6 Summary 61

III Security in CRTES 62

5 Attaining Side-Channel Attack Resiliency and Time-Predictability 63

5.1 Introduction 63
5.2 MBPTA and SCA Properties 64
5.2.1 Random Caches for MBPTA 64
5.2.2 Random Caches for Security 66

5.3 Assessing the Time-Predictability of Secure Cache Designs . . . 66
5.4 Assessing the SCA-robustness of Time-Predictable Cache Designs . . . 67
5.5 Time-Secure Caches . . . 69
5.6 Case Study . . . 71
5.6.1 Methodology . . . 71
5.6.2 Results . . . 72
5.7 Other Security Implications of Applying Randomization . . . 74
5.7.1 Power-Based Side-Channel Attacks . . . 75
5.7.2 Unauthorized Control Information Tampering (UCIT) . . . 75
5.7.3 Denial of Service (DoS) . . . 75
5.8 Related Work . . . 76
5.9 Summary . . . 76

IV Energy in CRTES **77**

6 Worst-Case Energy Consumption, a New Challenge **79**
6.1 Introduction . . . 79
6.2 Sources of Power Variability . . . 80
6.3 Current Practice on Processor-Level Typical and Maximum Power Estimation . . . 82
6.3.1 Validation and Verification . . . 84
6.4 Model-Based Task-Level WCEC . . . 84
6.4.1 Granularity and Accuracy . . . 85
6.4.2 Upper-Bounding the Activity Factor . . . 85
6.4.3 PAVT Variations . . . 87
6.5 Measurement-based WCEC Estimation . . . 88
6.5.1 Quality of the Measurements . . . 88
6.5.2 Input Space Coverage and Representativeness . . . 89
6.5.3 PAVT Variations . . . 91
6.6 Putting it All Together . . . 91
6.7 Related Work . . . 92
6.8 Summary . . . 93

7 Worst-Case Energy Consumption Modeling Methodology under the Presence of Process Variations **95**
7.1 Introduction . . . 95
7.2 PV-related power variability . . . 96
7.3 PV-aware Energy Modeling . . . 98
7.3.1 Random Nature of PV . . . 99
7.3.2 The Model . . . 99
7.3.3 Model Parameters . . . 100
7.4 WCEC Estimation Methodology . . . 101

7.4.1 WCEC Interpretation and Safety Standards 103
7.5 Experimental Results 103
7.5.1 Evaluation Framework 103
7.5.2 PV-generated power variability 104
7.5.3 Probabilistic WCEC Estimates 105
7.5.4 Multitask Workloads 106
7.5.5 Comparing EVT vs Gaussian approach 107
7.6 Summary 108

8 Detecting and Hampering Worst-Case Power Peak Events during Testing 109
8.1 Introduction 109
8.2 Challenges of Power Verification in Complex Processors 110
8.2.1 Power Delivery Network Sizing 110
8.2.2 Critical Real-Time Systems Verification 111
8.2.3 An Illustrative Example 112
8.3 Time-Randomization for Power Analysis 113
8.3.1 Event Alignment 113
8.3.2 An Illustrative Example 115
8.3.3 On Predicting Power Peaks 116
8.4 Quantitative Assessment 117
8.4.1 Experimental Setup 117
8.4.2 Power Verification 117
8.5 Related Work 119
8.6 Summary 119

V Conclusions and Future Work 122

9 Conclusions and Future Work 123
9.1 Impact 124
9.2 Future Work 125

Bibliography 127

List of Figures

1.1 Relation between memory layout and cache layout. 10

2.1 Nomenclature of possible execution times of a given task. 18
2.2 Timing analysis techniques taxonomy map. 19
2.3 Results from different WCET estimation techniques. 20
2.4 MBPTA process and example of pWCET curve. 24
2.5 Block diagram of a generic time-randomized cache. 26
2.6 Schematic of the hash logic of *hRP*. 26
2.7 Example of a 4-bit Benes network. 27

3.1 Simplified block diagram of the NGMP Architecture. 36
3.2 Overview of the evaluation framework 41
3.3 McPAT functional unit blocks nomenclature mapped to a detailed NGMP/LEON4 Architecture. 41

4.1 Example address distribution for a 16-set cache with default random modulo design. 47
4.2 Schematic of the baseline implementation of a random modulo cache. 48
4.3 Schematic of the *enhanced* implementation of a random modulo cache. 49
4.4 Example address distribution for a 16-set cache with the enhanced random modulo design. 50
4.5 IL1 per-set access distribution for `pntrch`. 53
4.6 DL1 per-set access distribution for `pntrch`. 54
4.7 Normalized increase in cache lifetime due to HCI impact. 56
4.8 Bit bias for the IL1 cache with *M* (top) and *ERM* (bottom) placement policies for the `pntrch` EEMBC benchmark. 58
4.9 Frequency spectrum of power dissipation for a regular processor (top) and a time-randomized processor (bottom). 59

5.1 (a) hRP and (b) ERM cache architectures. 68
5.2 Example of AUTOSAR application and seed management. 70
5.3 Time variations with respect to average across all different values of input byte number 4. 72
5.4 Effectiveness of the Bernstein's attack. 73

6.1 Average power dissipation of a program through execution time for different temperatures. 81
6.2 Usage of (measurement/analytical) models and measurements during the hardware design process. 83
6.3 Average energy consumption of a two-path program. 89
6.4 Diagram of the main challenges, and potential paths to follow, addressed by model-based and measurement-based WCEC estimation. . 90

7.1 Per-FUB power variability for the CPU and memory intensive benchmarks. 96
7.2 Example of different distributions per FUB. 98
7.3 WCEC estimation process and inputs it builds on. 102
7.4 Block diagram of our reference architecture. 104
7.5 pWCEC distribution in μJoules and empirical CCDF of the PV-adjusted energy measurement. 106
7.6 pWCEC estimates for multi-core workloads. 107
7.7 pWCEC curves and ECCDF of the samples. 108

8.1 Histogram of instantaneous power measurements in a simple and a complex processor. 111
8.2 Power profile on a conventional (simple) architecture when running two unsynchronized benchmarks. 112
8.3 Power profile on a conventional (simple) architecture when running two synchronized benchmarks. 112
8.4 Power profile on a conventional (complex) architecture when running four benchmarks. 113
8.5 Power profile on a time-randomized (simple) architecture when running two unsynchronized benchmarks. 114
8.6 Power profile on a time-randomized (simple) architecture when running two synchronized benchmarks. 114
8.7 Power profile on a time-randomized (complex) architecture when running four benchmarks. 115
8.8 Power dissipation over time of 4 different EEMBC in a randomized hardware. 118
8.9 Probabilistic curve and empirical sample of power dissipation values (in μW) for benchmark `aifirf`. 118

List of Tables

3.1 Summary of modeled reference architectures. 37
3.2 Summary of benchmarks used for evaluation. 40

4.1 Summary of the different random placement implementations and its trade-offs. 51
4.2 Distribution of accesses across sets for the different placement functions and L1 caches. 52
4.3 Distribution of accesses across sets for the different placement functions in the L2 cache. 55

6.1 Toggle coverage for different Workloads on a RTL model of the LEON3. 87

7.1 FUB power variability. 97
7.2 Summary of distributions used for modeling PV features. 99
7.3 Parameters needed for applying the methodology. 101
7.4 Maximum observed energy, and pWCEC (in μJ) with PV. 105

8.1 Maximum peak count for the deterministic and randomized architectures, and probabilistically estimated number of power peaks 120

List of Abbreviations

AD	Autonomous Driving.
AES	Advanced Encryption Standard.
AI	Artificial Intelligence.
AMBA	Advanced Microcontroller Bus Architecture.
AP	Analysis Phase.
ATS	Automotive Systems.
BTI	Bias Transistor Instability.
CABA	Cycle-Accurate Bit-Accurate.
CCDF	Complementary Cumulative Distribution Function.
COTS	Commercial Off-The-Shelf.
CRTES	Critical-Real Time Embedded Systems.
CV	Coefficient of Variation.
DoS	Denial of Service.
DSR	Dynamic Software Randomization.
DVFS	Dynamic Voltage and Frequency Scaling.
ECCDF	Empirical Complementary Cumulative Distribution Function.
ECU	Electronic Control Unit.
EEMBC	Embedded Microprocessor Benchmark Consortium.
ERM	Enhanced Random Modulo.
ESA	European Space Agency.
EVT	Extreme Value Theory.
FFT	Fast Fourier Transform.
FPU	Floating-Point Unit.
FUB	Functional Unit Block.

GPGPU	General Purpose Graphics Processing Unit.
HCI	Hot-Carrier Injection.
HDTA	Hybrid-Deterministic Timing Analysis.
HPTA	Hybrid-Probabilistic Timing Analysis.
hRP	Hash-Based Random Placement.
I.I.D.	Independence and Identical Distribution.
IFU	Instruction Fetch Unit.
ILP	Integer Linear Programming.
IMA	Integrated Modular Avionics.
IoT	Internet of Things.
IP	Intellectual Property.
IPET	Implicit Path Enumeration Technique.
ISA	Instruction Set Architecture.
LRU	Least Recently Used.
LSU	Load Store Unit.
M	Modulo.
MBDTA	Measurement-Based Deterministic Timing Analysis.
MBPTA	Measurement-Based Probabilistic Timing Analysis.
MBPTA-CV	Measurement-Based Probabilistic Timing Analysis using Coefficient of Variation.
MOET	Maximum Observed Execution Time.
NBTI	Negative-Bias Transistor Instability.
NGMP	Next Generation Microprocessor.
NMOS	N-type Metal-Oxide-Semiconductor Logic.
NoC	Network-on-Chip.
OP	Operation Phase.
OTA	Over-the-Air.
PAVT	Process, Aging, Voltage and Temperature.
PBTI	Positive-Bias Transistor Instability.
PDF	Probability Distribution Function.

PDN	Power Delivery Network.
PMC	Performance Monitoring Counter.
PMOS	P-type Metal-Oxide-Semiconductor Logic.
PMU	Performance Monitoring Unit.
PRNG	Pseudo-Random Number Generator.
PV	Process Variation.
pWCEC	Probabilistic Worst-Case Energy Consumption.
pWCET	Probabilistic Worst-Case Execution Time.
RM	Random Modulo.
RPV	Relative Power Variability.
RTL	Register-Transfer Level.
RTOS	Real-Time Operating System.
RVN	Resonant Voltage Noise.
SCA	Cache-Timing Side-Channel Attacks.
SCS	Safety-Critical Systems.
SDTA	Static-Deterministic Timing Analysis.
SEU	Single-Event Upset.
SoC	System-on-Chip.
SPTA	Static-Probabilistic Timing Analysis.
SSR	Static Software Randomization.
SWC	Software Component.
TDDB	Time-Dependent Dielectric Breakdown.
TDMA	Time-Division Multiple Access.
TDP	Thermal Design Point.
TRP	Time-Randomized Processors.
TSCache	Time-Predictable Secure Cache.
UCIT	Unauthorized Control Information Tampering.
V2V	Vehicle-to-Vehicle.
V&V	Validation and Verification.
WCEC	Worst-Case Energy Consumption.
WCET	Worst-Case Execution Time.

Part I

Introduction

Chapter 1

Introduction

> *"Safety? Where the fuck's that? Her aunt in the Eyrie is dead. Her mother's dead. Her father's dead. Her brother's dead. Winterfell is a pile of rubble. There is no safety, you dumb bitch."*
>
> — Sandor Clegane

Historically, the use of computers has been restricted to desktop systems and large platforms such as mainframes and supercomputers. As the chip manufacturing and integration technology improved over the years, computing systems have benefited from lighter and smaller form factors which allowed embedding them into devices that required spatial mobility and small integration sizes. Embedded Systems can be found in a variety of devices, from simple household appliances to Electronic Control Unit (ECU) inside cars. The special properties of Embedded Systems have made them more numerous than traditional computing platforms and their growth in market share [1] and applicability is projected to increase even more in the following years.

The unprecedented growth in popularity, coupled with their low cost and a higher reliability than their mechanical counterparts, has lead Embedded Systems to be integrated into devices that perform functionalities where highly valuable material or even lives are at stake. Space, avionics, automotive and railway systems are just some examples of critical domains where Embedded Systems are used to perform critical functionalities while increasing the integration scale of the system. Those are generally referred to as Critical-Real Time Embedded Systems (CRTES).

1.1 A Changing Paradigm in CRTES

Safety standards guide the planning, design, development and validation processes that CRTES manufacturers should follow to attain certification[1]. By following this

[1] Certification is the process of determining the safety goals of the system, specifying the safety requirements, designing the system in accordance with those requirements, verifying the system against those requirements, and validating that those requirements, and ultimately also the safety goals, are met during the integration of the different items of the system [2]. Certification is issued by legally stated certification authorities and allows the operation and deployment of certain products.

guidance, CRTES designers can easily provide proof and guarantee CRTES correctness to certification authorities. In the case of CRTES, manufacturers and system integrators need to provide evidence that both functional and non-functional safety requirements are met. For instance, certification standards like ISO-26262 [3] for automotive or DO-178C [4] for avionics provide the guidelines and recommendations that vehicular systems should follow to comply with safety requirements. These guarantees are provided by thorough Validation and Verification (V&V).

Software timing is a long-studied non-functional property of CRTES that must be taken into account during the V&V process. Ensuring timing correctness requires gathering evidence that software tasks are completed before their deadlines and therefore that the system is able to guarantee a feasible schedule. According to the system under analysis, different levels of criticality might imply varying degrees of evidence of correctness that must be met according to the specific properties defined in the standard.

In general, V&V is covered with extensive testing campaigns that increase production costs due to the involvement of many qualified experts probing and validating the designs for a high degree of confidence. In some cases the validation process might even take 50% of the development budget [5].

Up until recently, V&V has been kept under feasible constraints [5] due to the simplicity of the past CRTES. This past simple CRTES were typically designed as *federated architectures* where different software functionalities were kept isolated from each other in its own ECU [6].

However, recently CRTES are experiencing important changes due to the increasing demand for more advanced functionalities that pushes manufacturers to gain the competitive edge in the market. Concepts like Integrated Modular Avionics (IMA) [7, 8] were among the first steps of the CRTES evolution by proposing the move from federated architectures to more integrated ones and involving the use of shared resources to diminish the costs of CRTES. Nowadays, new software applications are driving CRTES evolution. Artificial Intelligence (AI) algorithms have gained momentum thanks to their improvements in speed and accuracy. This technological breakthrough becomes fundamental to applications such as Autonomous Driving (AD), currently one of the most important growing industry paradigms [9, 10], and forces further changes to CRTES as its software becomes more demanding.

Emerging CRTES software applications (e.g., AD) demand performance well beyond what typical CRTES processors can offer. In the case of AI, the amount of data and computation is scaled to unprecedented levels never seen in the embedded domain and requires complex high performance processors and hardware features. Some industrial projections even indicate that the compute performance needed in vehicles will increase in 100x in the following years [11]. This evidences that performance will be of paramount importance for future CRTES.

In order to provide high computing performance, modern systems adopt a plethora of high performance hardware features such as multi-cores, hardware accelerators and other hardware optimizations. For instance, cache memories allow data to be accessed much faster by keeping a subset of the main memory closer to the processor. In the same way, multi-cores integrate the performance of multiple processors by sharing some common arbitrated structure to perform multiple tasks at once without fully replicating all resources. All of these changes have a substantial impact in the V&V process.

The benefit of introducing high performance hardware features comes at the expense of more complexity. Among these features, we find shared functional units, speculation schemes, caches, and many more elements that increase system complexity. This increase in complexity hampers V&V of functional and non-functional metrics and induces a cost increase that in some cases might even make unfeasible the adoption of new technology.

The introduction of this new hardware high-performance features brings a whole set of new properties that makes other non-functional aspects arise in importance. Besides hampering timing V&V, modern and future CRTES based on complex hardware and software challenge the achievement of other requirements like reliability, security and energy.

1.2 Emerging Non-functional Requirements

Non-functional metrics comprise all those aspects of the system design that lay outside the correctness of the (functional) computational operation. In the case of CRTES, extensive V&V is also required for all those metrics. Four main non-functional metrics are covered in this thesis: Software timing (introduced in previous sections), hardware reliability, security and energy.

1.2.1 Software Timing

In CRTES, typically the major non-functional concern has been to provide guarantees that tasks execute in their budgeted amount of time and that they are able to meet their deadlines. Therefore, accuracy and tightness are of paramount importance when analyzing the execution time of tasks. To that end, during system planning and design phase the Worst-Case Execution Time (WCET) is estimated and used to adequately size the platforms and provide feasible schedules. The process of deriving WCETs relies on the ability of experts to precisely exert control and account for the worst possible states and instruction sequences that a processor might exhibit at deployment time. Up until now, either static analysis techniques, through their processor models, or measurement-based techniques, through their engineering margin factors[2] [12] were enough to derive WCETs feasibly.

However, these currently existing techniques might have a hard time coping with all the different states in which a machine can be, due to the introduction of new complex high performance hardware features which require even more in-depth knowledge of the processor and increase timing uncertainty. Furthermore, manufacturers usually protect their Intellectual Property (IP) by hiding and obfuscating details of the

[2] A common typical margin factor would be 20% for a single core processor in domains like avionics or space.

architecture in manuals which greatly jeopardizes the possibility of understanding all the internals of their platforms [13].

This scenario calls for new timing analysis techniques that are able to handle the exponential growth of processor states and provide trustworthy timing estimates without delivering too much pessimism.

1.2.2 Hardware Reliability

Shrinking the device size allows the integration of more transistors, wires and vias in the same space boosting performance by allowing more cores on the same package, higher operating frequencies, lower power dissipation and other improvements. Despite all the advantages, shrinking devices makes processors less reliable. For instance, shrinking the device size has a negative impact in the aging of the devices which directly clashes with the needs of upcoming CRTES [14, 15, 16]. Some of the main sources of failure due to aging relate to degradation of (1) different parts of the transistor caused due to Hot-Carrier Injection (HCI), Bias Transistor Instability (BTI), either negative (NBTI) or positive (PBTI) and Time-Dependent Dielectric Breakdown (TDDB); (2) different parts of wires due to electromigration and stress migration; and (3) the pins and packaging itself due to thermal cycling.

The expansion in the use of smaller technologies is leading them to be deployed in systems for which their expected lifetime is more extent than typical consumer devices [17, 18]. Personal desktop and laptop computers or smartphones have relatively low expected lifetime in the 5 to 10 years time frame. On the other hand CRTES deployed in cars or satellites are expected to last for much longer time spans [19].

In some domains, CRTES are deployed in environments that constantly stress the reliability capabilities of their hardware. This is the case for avionics and space, where the presence of radiation and cosmic rays is more prevalent than at surface level. CRTES in this kind of domains are more likely to be affected by Single-Event Upset (SEU) and experience soft-errors caused by the aforementioned phenomena. In the presence of critical funcionality, measures must be put in place to prevent catastrofic failures due to the presence of SEU.

Further more, reliability failures do not only affect the functional correctness of the device since in the context of fault-tolerant processors, failures will impact processors performance and therefore the WCET further compromising the timing analysis process [20, 21, 22]. For instance caches are processor features that improve memory access time. Over time, hardware components of caches degrade and some bits might turn unusable diminishing the effective size of the cache. This size reduction translates into less capacity to store data in the cache and therefore, into more cache misses which increase the execution time of tasks [23]. Moreover, this timing impact is arbitrary and potentially inordinate thus being hard to account for. This effect is an example of how the initial analysis of WCET might be compromised by faults and hardware reliability as the devices show different behaviors from production to deployment and until the end of their lifespan.

1.2.3 Security

Security violations are one of the most important research fields nowadays due to their impact in society. It is estimated that the cost of information attacks only in the United States amounted between 57 and 109 billion dollars just in the year 2016 [24]. As society becomes more connected and computerized it is expected to increase even more as more information and control is exposed to the environment. Additionally, the security domain turns out to be one of the most challenging, not only because it demands profound understanding of the mathematics and cryptographic algorithms involved, but also because of the variety of attack vectors and hidden vulnerabilities in hardware designs. This specific characteristic unique to the security domain makes it very hard to predict and evaluate whether systems are secure or not and would require very complex models for which it is not completely known if the model is suitable for attacks yet to discover.

Communication is a basic feature of modern computers and CRTES are not an exception. Communication in vehicles allows information sharing that enables devices to coordinate and acquire information out of the reach of their sensors. For instance, Vehicle-to-Vehicle (V2V) communication or swarm intelligence are new concepts arising in the vehicle domain that will require open communication between vehicles and the environment [25]. Moreover, the new wireless connection technologies like 5G will enable Over-the-Air (OTA) software updates of vehicles in order to decrease vehicle maintenance costs [26, 27]. At the same time, the introduction of the much needed high performance hardware features often involves the presence of shared resources and performance improving features. The conjunction of the new connectivity capabilities with the new hardware resources opens the door to new attack vectors and covert information channels that malicious agents can use to steal information or take control of CRTES.

In the case of CRTES this problem is aggravated with the context in which such systems operate. In this domain, users do not only might loose control of their information, but highly costly equipment and even human lives are at stake if an attack successfully manages to take control over critical equipment (e.g., vehicles).

1.2.4 Energy

The proliferation of battery-powered Internet of Things (IoT) and power-constrained devices controlling increasingly critical aspects of human life is relentless in domains such as health, smart cities, and intelligent transportation systems [28]. The complexity of software running on those devices increases every generation to cover the demands for more autonomous operation, implementing decision making and data analysis techniques among others. Handheld devices, which will govern part of the critical-applications functionality, will also inherit part of application criticality.

In battery-powered devices, energy is one of the most important resources as battery life is a key element for products' competitive edge. Analogously, power-constrained devices cannot exceed specific energy thresholds in short time frames due to limited power sources (e.g., solar cells in space). Therefore, when the device implements some type of critical functionality, a new set of energy-related requirements arises. This emanates from the fact that critical functionality (and in particular the hardware and software implementing it) has to undergo a stringent V&V process to show adherence to the prospects in domain-specific standards [29]. With the increase in complexity and performance of new CRTES, energy demands will put more pressure on the battery capacity making worst-case energy estimation of paramount importance. Similarly, power dissipation profiles under worst conditions becomes hard to predict, hence challenging the design for power-constrained devices and their power delivery networks.

Energy consumption impacts CRTES in many significant ways. For instance, thermal effects are directly related to the power dissipation of electric devices. Many CRTES are usually located in confined spaces that difficult air circulation and temperature dissipation and sometimes perform tasks under extreme heat conditions [18, 30]. The weight limitations also place important restrictions on the capabilities of dissipation systems that can be installed, hence the amount of consumed energy by CRTES places constraints in their design. Additionally, in order to keep temperature within specific operating ranges, some processors use frequency throttling or might shut-down some parts of the device. Voltage or frequency scaling during operation jeopardizes timing analysis done during the design phase leading to manufacturers disabling those features and further limiting the performance capabilities. All these implications exacerbate the need for energy and power accounting under the worst conditions as they need to show adherence to stringent power and energy bounds.

1.3 Summary of Challenges in future CRTES

As computing performance demands rise in CRTES, new hardware and software features are introduced to meet such demands. This results in CRTES increased complexity that, in turn, requires more effort to successfully carry out V&V. Historically, CRTES have been kept very simple to ease in the validation efforts and reduce its sky rocketing costs [5]. However, in order to keep the competitive edge of their products, CRTES manufacturers desperately need the high performance provided by advanced hardware and software. Hence they are faced with the conundrum of providing high performance while containing V&V costs.

Moreover, CRTES not only require an increase in computing performance, but also to be integrated with technologies that extend their past capabilities. To appeal to the future markets and costumers, CRTES manufacturers must integrate demands like communication and internet connection while providing security against external threats and at the same time attaining highly reliable and durable devices. Introducing all these requirements greatly increases the effort needed to validate and verify CRTES in order to guarantee correct functional and non-functional behavior. Future CRTES need all these requirements to be tackled at the same time even when some of them might (initially) be opposed.

1.4 Time-Randomized Processor Architectures

To approach the need for timing validation, randomization has recently been introduced in CRTES to obtain (probabilistic) timing predictability in complex processors [31]. Many new processor features introduce new processor states that impact execution time in arbitrary ways. By introducing randomization the dependence and relationship between the specific processor states and execution times is broken. The rationale behind this idea is that injecting randomization into the timing behavior of hardware resources that exhibit time jitter will give probabilistic properties to time variability and make the system probabilistically analyzable [32]. By injecting randomization in certain variable latency hardware features a new class of processors (Time-Randomized Processors (TRP)) arises, with the capability of effectively randomizing the execution time of tasks.

Figure 1.1 showcases how access time to memory in TRP is randomized. Caches are fast memories used to quickly feed data into processors functional units. However, they only allow a subset of data from main memory to be stored at any given time. Placement and replacement policies are then implemented to determine which data should be kept and which data should be evicted. The execution time of tasks is therefore dependent on the specific placement policy since finding the desired data in cache (hit) translates into faster access times, while not finding it (miss) translates into longer execution times. Traditionally, the placement of data is deterministically determined by the address in memory of the data. Therefore the memory layout defines deterministically the cache layout, the miss and hit ratio and consequently the execution time. Different memory layouts have an arbitrary impact in the execution time. Typically during WCET estimation of each individual functionality the final memory layout of the integrated system is unknown, hence uncertainty further hardens WCET estimation . Instead, in a TRP the placement is randomized using a *random seed* that can be changed. Now the hit and miss patterns are randomly determined and vary independently from the memory placement. Therefore, measurements for a given functionality before integration are probabilistically representative of its behavior after integration. Figure 1.1 shows how a single static memory mapping can create multiple different and independent cache memory mappings by the use of different random seeds and smart placement policies. Different cache mappings will have different conflicts in cache, exhibiting different execution times for a particular task and effectively randomizing the execution time of memory accesses. Hence, by collecting a sufficiently large number of measurements, cache impact in execution time can be characterized.

By acquiring randomized properties, measurements of tasks' execution time can now be treated as independent observations of the complete possible execution time space [33]. A process of sampling is used then to evaluate possible processor states through its execution time. Because of the probabilistic nature of the randomized behavior, execution times will be observed according to a given probability, average cases will be observed more frequently and extreme cases will have a lower probability of appearance. Now, if enough observations are gathered, a mathematical argument

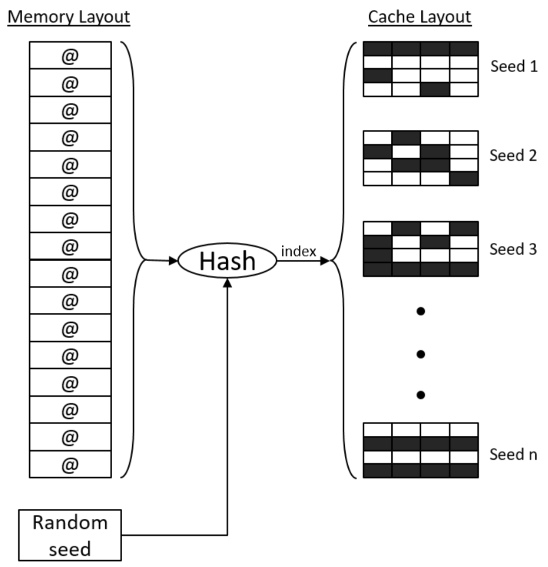


Figure 1.1: Relation between memory layout and cache layout. TRP randomize cache layouts to explore extreme execution times of tasks. Black tiles show conflicts that create evictions and misses in different cache layouts.

can be made about having observed extreme events with a certain probability, similarly to performing a Monte Carlo experiment. Despite the usefulness of statistical exploration, the observability of worst cases is limited to the amount of samples. To develop further guarantees, tools are needed to augment the statistical analysis and extrapolate the results. Coupling the sampling process with statistical tools like Extreme Value Theory (EVT) allows end-users to seamlessly derive WCET without the need of extensive knowledge of the processor and backed up by mathematical probabilistic guarantees [13]. EVT is a branch of statistics that provides the tools to extrapolate extreme values from a small amount of samples basically allowing to extrapolate the execution times that would be observed if millions of samples were taken by just having thousands of samples. Measurement-Based Probabilistic Timing Analysis (MBPTA) [34, 33] is the process of applying this specific methodology which allows to estimate the WCET of a task by injecting randomization into certain processor features and then sampling and mathematically treating the results. The manifold advantages that randomization (within MBPTA methodology) provides with respect to other timing-analysis techniques makes TRP a suitable solution for providing the much needed high performance without loosing the timing guarantees that CRTES must provide [35, 36].

1.5 Hypothesis

Despite the suitability of TRP for tackling the software timing problem, new applications bring other challenges to CRTES such as energy, security and reliability constrains, but the potential benefits of TRP in solving them remain to be assessed.

Moreover, the new challenges affecting the non-functional properties of CRTES must be met simultaneously together with the old ones, even when some are opposite requirements. For instance, verifying timing predictability usually requires observability and transparency while providing evidence of security requires obfuscation and unpredictability to provide protection against attacks. At the same time, providing performance improvements like shrinking the device size might come at the cost of diminished reliability.

Improving all metrics is the key to the success of new technologies in a highly demanding environment like the CRTES domain. Up until now, randomization has been successfully used to jointly tackle what initially seemed to be two contradicting principles, the need for more powerful hardware, and the need for timing predictability [13]. In that case, TRP, by introducing randomization in software timing, simplifies the use of probabilistic techniques to estimate with high confidence and accuracy the timing behavior of performance enhancing hardware features that usually exhibit hard to predict timing variability.

Randomization provides mathematical properties widely exploited in many domains. Randomized events follow distributions, break dependencies and meddle with repeatability. All these effects can be used to tackle the challenges of CRTES. For instance, randomized distributions have been widely used in mathematics to predict patterns that at first instance seem chaotic. Timing and energy models with a large number of variables, like in complex processors, are daunting and costly to operate and verify, but by introducing randomization, processor behavior can be modeled in an easier, more accurate and effective way. Furthermore, randomization also breaks dependencies and repeatability, properties needed typically to reliably transmit information but become undesirable under cases were non-disclosure of the information or control has to be preserved, like modern CRTES with specific security requirements. Moreover, decay and degradation are effects much related to usage. Under deterministic operation, these effects are greatly amplified, but when under randomization schemes random distributions allow for smoother and less acute usage patterns that can increase lifetime and reliability of devices. Overall, this thesis holds and tries to verify the hypothesis that injecting randomization in the timing behavior of hardware resources, not only benefits the already studied non-functional metrics (i.e., timing-analyzability), but its properties can also be used to impact positively on all the other non-functional metrics. Therefore, the question we try to address is if randomization can be used to tackle the challenges posed by the new hardware reliability, security and energy requirements at the same time in CRTES.

1.6 Contributions

The goal of this thesis is to pioneer in the use and application of TRP for non-functional requirements outside the timing-analysis for which TRP were initially proposed for. This thesis sets the fundamental knowledge for using TRP as a solution to tackle the new challenges posed by high performance hardware and software features on CRTES design, testing, implementation and usage.

The thesis is presented into three different themes each referring to one of the non-functional metrics explored, reliability, security and energy. The vastness and richness of topics inside each of these domains makes unfeasible tackling them to its completeness so this thesis attacks specific challenges under each theme. The following sections specify the contributions of this thesis and narrow down which particular topics of the specified themes the thesis covers.

1.6.1 Reliability

This thesis first tackles the reliability domain, specifically, hardware reliability under the presence of aging effects. As a first contribution to this theme, we analyze the impact of a new randomized cache placement policy (Random Modulo) [37] in terms of HCI [38] aging and BTI [39] for L1 caches as we hypothesize that randomization will show greater results and leave other effects like electromigration or TDDB for future work. Our results show that benefits in terms of HCI are meaningful and can be further improved, whereas BTI gains are very limited. Secondly, we propose an enhanced randomized cache placement design (Enhanced Random Modulo) to mitigate HCI aging. Finally, we analyze the impact of an existing randomized cache policy (Hash Random Placement) in terms of HCI in L2 caches showing that benefits are already significant. Additionally, we also contribute towards demonstrating the resilience of TRP against resonant voltage noise by analyzing patterns in the energy consumption behavior.

1.6.2 Security

In the security domain we focus mainly on a particular type of security intrusions due to the implications of time-randomization, Cache-Timing Side-Channel Attacks (SCA) [40] and how a Timing-Analyzable processor can be made resilient against this kind of threats. We first make an in-depth analysis of the properties required to enable MBPTA to deal with jittery timing behavior of applications running on complex hardware. We cover how randomization helps dealing with SCA and describe the vulnerability of the randomization support for MBPTA to specific SCA. We proceed by assessing the time predictability of the randomization support for SCA solutions showing that they fail to meet MBPTA principles. We then propose a Time-Predictable Secure Cache (TSCache) that provides increased resilience in front of specific SCA while keeping MBPTA compliance. The TSCache, hence, reconciles security (robustness against certain SCA) and safety (by adhering to MBPTA principles to derive reliable timing budgets) in cache design. Finally, with our simulator tool modeling a commercial automotive processor, we experimentally show the resilience of our solution against the Bernstein attack [41] while keeping MBPTA compliance. Other security challenges like Denial of Service (DoS) attacks and other vulnerabilities are also assessed to how can those be tackled with randomization.

1.6.3 Energy

The contributions in the energy domain focus on the topic of Worst-Case Energy Consumption (WCEC) and power modeling/estimation and validation. Our first contribution is a taxonomy of the factors affecting dynamic and static energy consumption and hence, WCEC estimation. We describe the difficulties in deriving tight WCEC estimates using model and measurement-based approaches. We capitalize on how Process, Aging, Voltage and Temperature (PAVT) variations cause jitter on power with bearing consequences on WCEC estimation. Overall, we settle the ground on the challenges for practical and reliable WCEC estimation and aim at becoming a reference for future works on WCEC estimation.

Our second contribution builds on the previous concepts to propose a model to derive tight WCEC estimates in the presence of Process Variation (PV), which emanate from the fact that PV causes energy consumption variations and, therefore, different WCEC across different nominally-identical processor units. Our methodology, performs the entire estimation on a single processor unit, while delivering WCEC estimates that hold for all processor units and simplify the V&V process of CRTES.

As third contribution, we tackle pathological systematic worst-case power dissipation patterns that may remain hidden during testing and occur during operation. We show how time-deterministic behavior of processors challenges, in general, identifying whether power measurements in the test campaigns expose relevant power peaks. We demonstrate how the use of TRP contributes to exposing feasible power peaks and to the mitigation of pathological power dissipation effects, and, therefore, enables the ability to provide safer and tighter guarantees.

1.7 Structure of the Thesis

The contents of each of the chapters is as follows:

- Chapter 2 presents background concepts and terminology of the CRTES domain and timing analysis. Also, some background on hardware reliability, security and energy is also presented together with the description of available randomized hardware solutions for MBPTA.
- Chapter 3 explains the experimental setup, methodology and the tools we use to quantify and evaluate the thesis' hypothesis.
- Chapter 4 tackles reliability in CRTES. It introduces improvements to existing randomized hardware designs to improve CRTES lifetime and assess the impact of TRP in the reliability domain.
- Chapter 5 describes how protection can be achieved against security intrusions and what role do TRP play in securing CRTES. It also demonstrates TRP effectiveness when protecting CRTES with a real malicious attack use case.
- Chapter 6 opens the energy theme by surveying and analyzing the current state of Worst-Case Energy Consumption estimation, stating the challenges of obtaining safe WCEC and proposing a taxonomy for energy estimation methods.

- Chapter 7 proposes a novel methodology for obtaining safe WCEC estimates under the presence of PV that appear during processor manufacturing.
- Chapter 8 proposes the use of TRP to discover, account and hinder pathological power peaks in CRTES processors. First we explain how with randomized hardware events of high power demand can be observed while with deterministic it would require a significant effort and expert knowledge to expose them. We follow up with a demonstration of how these pathological scenarios are naturally mitigated by randomized architectures.
- Chapter 9 presents the final conclusions of the thesis, defines its impact and exposes directions for future work.

1.8 List of Publications

General

1. **Randomization for Safer, more Reliable and Secure High-Performance Automotive Processors.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *In IEEE Design & Test (D&T).* Volume 36, Issue 6, 39-47; July 2019. DOI: 10.1109/MDAT.2019.2927373

Reliability

2. **Aging Assessment and Design Enhancement of Randomized Cache Memories.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *In IEEE Transactions on Device and Materials Reliability (TMDR).* Volume 17, Issue 1, 32-41; March 2017. DOI: 10.1109/TDMR.2017.2654548
3. **Resilient Random Module Cache Memories for Probabilistically-Analyzable Real-Time Systems.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *In IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS).* Sant Feliu de Guíxols, Spain; July 2016. DOI: 10.1109/IOLTS.2016.7604666

Security

4. **Cache Side-Channel Attacks and Time-Predictability in High-Performance Critical Real-Time Systems.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *In ACM/ESDA/IEEE Design Automation Conference (DAC).* San Francisco, USA; June 2018. DOI: 10.1109/DAC.2018.8465919. Best Paper Award Nominee.

Energy & Power

5. **Worst-Case Energy Consumption: A New Challenge for Battery-Powered Critical Devices.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *In IEEE Transactions on Sustainable Computing (TSUSC).* September 2019. DOI: 10.1109/TSUSC.2019.2943142

6. **An Approach for Detecting Power Peaks during Testing and Breaking Systematic Pathological Behavior.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *In Euromicro Conference on Digital System Design (DSD).* Kallithea, Greece; August 2019. DOI: 10.1109/DSD.2019.00083
7. **Modeling the Impact of Process Variations in Worst-Case Energy Consumption Estimation.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *In Euromicro Conference on Digital System Design (DSD).* Kallithea, Greece; August 2019. DOI: 10.1109/DSD.2019.00092

1.8.1 Informal Publications

8. **Four Birds with One Stone: On the Use of Time-Randomized Processors and Probabilistic Analysis to Address Timing, Reliability, Energy and Security in Critical Embedded Autonomous Systems.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *Workshop on Energy-Secure System Architectures.* McLean, USA; May, 2019.
9. **On the Suitability of Time-Randomized Processors for Secure and Reliable High-Performance Computing.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *Barcelona Supercomputing Center Severo Ochoa Doctoral Symposium.* Barcelona, Spain; May, 2017.
10. **Time-Randomized Processors for Secure and Reliable High-Performance Computing.** D. Trilla, C. Hernández, J. Abella and F.J. Cazorla. *Workshop on Pioneering Processor Paradigms.* Austin, USA; February 2017.

Chapter 2

Background

> *"It is of great advantage to the student of any subject to read the original memoirs on that subject, for science is always most completely assimilated when it is in the nascent state."*
>
> — James Clerk Maxwell

2.1 Timing Analysis

Safety standards provide guidance to Critical-Real Time Embedded Systems (CRTES) industry on the evidence to provide on systems' correct behavior to pass the certification process. To that end they define the Validation and Verification (V&V) process that CRTES must undergo on their functional and non-functional behaviors. *Software Timing* is the main non-functional requirement to be satisfied by CRTES. The software timing analysis process aims to provide guarantees on estimated execution times so tasks in CRTES can meet their deadlines and produce feasible schedules. Timing analysis builds primarily on the ability to derive trustworthy Worst-Case Execution Time (WCET) estimates and scheduling. This activity is hampered as the execution time of a task, and therefore its WCET, depends on many factors that range from the number of instructions being executed and the input data to the underlying hardware microarchitecture. This vastness in parameter diversity directly impacts the execution time, which eventually leads to jitter (variation) in execution times, even when running the same exact binary. For instance, in the case of microarchitectural impact, cache memories rely on temporal and spatial locality of data to provide performance improvements, which means that successive runs will benefit from data reuse, hence reducing their execution time. In this line, Figure 2.1 illustrates the common result from repeatedly measuring the execution time of a program executed in the same hardware.

In Figure 2.1 we can identify three distinct relevant points. From lowest to highest execution time, first we have the Maximum Observed Execution Time (MOET), this defines the highest water mark and it is obtained through an extensive testing campaign with multiple inputs and system states. Following MOET is the real WCET.

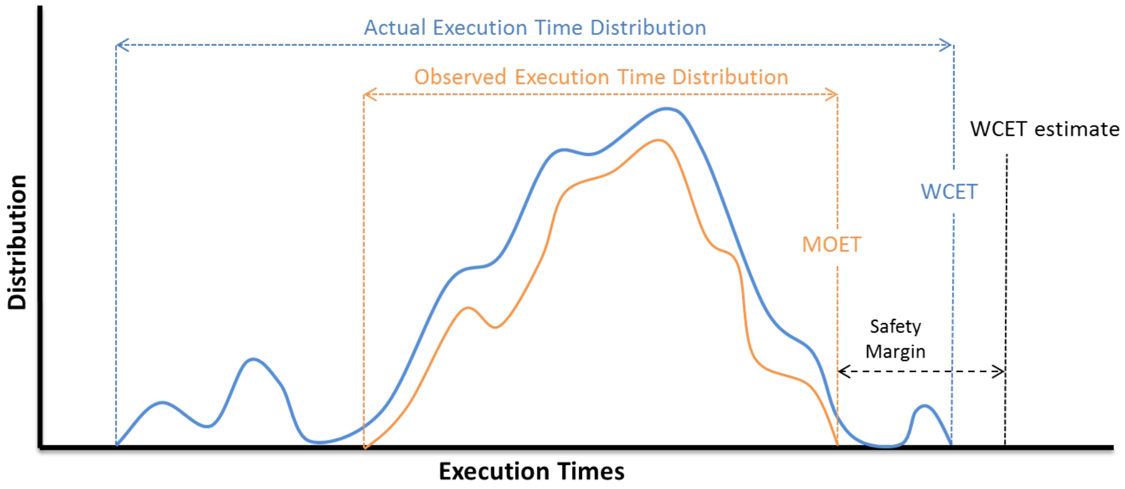


Figure 2.1: Nomenclature of possible execution times of a given task. Repeatedly sampling the execution time of a task will show a distribution of observed execution times. The actual distribution is generally unknown, but the goal of computing the WCET estimate is to obtain an upper-bound to the worst-case execution time.

In an ideal scenario, which implies total knowledge of the software and hardware, experts could make the MOET match the real WCET by managing inputs and processor states so the worst possible case for that particular task and system is exposed. Finally, we have the estimated WCET. The main goal of a timing analysis technique is to approximate the estimated WCET to the real WCET, but due to uncertainty a safety margin is usually applied. The safety margin width defines the overestimation of the timing analysis technique and should be as tight as possible without compromising safety. The consequence of an overestimated WCET is an overdesigned system that greatly increases the procurement and integration costs of CRTES. Overall, the main challenges lay on providing evidence that the derived WCET estimate is trustworthy while minimizing overestimation.

To tackle this problem, the research in this field has led to a myriad of techniques [32, 42] that cope with such a vast problem. We proceed to provide a brief description of the general taxonomy of methodologies and techniques that govern the *Timing Analysis* field of research (see Figure 2.2). In that regard, we can separate the *Timing Analysis* techniques in two different categories: Deterministic and Probabilistic methodologies.

2.1.1 Deterministic Timing Analysis

Deterministic methods seek for a single WCET value that holds as an upper-bound on any possible existing execution time.

- **Static-Deterministic Timing Analysis (SDTA)** [43]. SDTA techniques derive WCET estimates without executing the program under analysis and employ instead timing models of the hardware and software. Applying SDTA not only

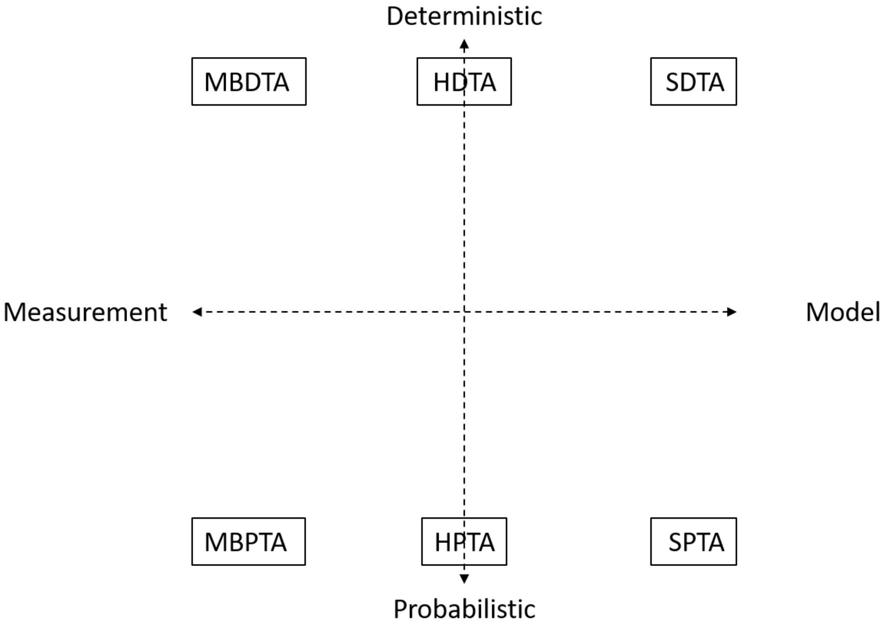


Figure 2.2: Timing analysis techniques taxonomy map. Mainly classified into probabilistic and deterministic or measurement-based and model-based techniques, although hybrid versions exist in both axes.

requires analysis of the program binary but also of the hardware architecture in which it will be executed. The basic methodology involves applying techniques like Implicit Path Enumeration Technique (IPET) [44] to determine worst cases of program paths, and use the acquired knowledge together with detailed timing architectural models to derive a single WCET. The trustworthiness of SDTA techniques resides on the validity of the underlying model used to compute the latency (or cost) of each instruction and processor event. As a downside, despite its maturity, as processor complexity rises, providing evidence of the reliability of static timing models is becoming increasingly difficult [13] and even causes an slow-down in the process of integrating greater performance in CRTES. Additionally, problems of tightness to leverage hardware and software complexity also arise.

- **Measurement-Based Deterministic Timing Analysis (MBDTA)** [45]. MBDTA techniques are widely used in industry due to its simplicity and low cost. The basic working principle is to collect measurements under stressing conditions of the task under analysis and apply a safety margin to the MOET to cover unknowns. In MBDTA techniques, the responsibility for control of state and input conditions is left to the end user. Being able to derive the correct analysis conditions that upper-bound conditions at operation is the key aspect for obtaining a reliable measurement-based WCET. This approach is also hindered by the increasing processor complexity, due to all the new processor states that must be taken into account.

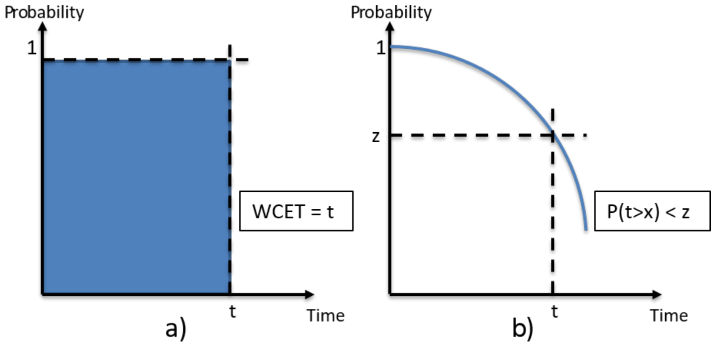


Figure 2.3: Results from different WCET estimation techniques. a) shows results from SDTA. b) depicts probabilistic estimates of execution time from the application of MBPTA. Note that the probability axis in b) shows the complementary cumulative density function of execution times.

- **Hybrid-Deterministic Timing Analysis (HDTA)** [46]. Hybrid techniques use a combination of the previously mentioned methods. Some involve the use of MBDTA to derive architectural details like latency when they are not publicly available to later use those with a static analysis tool. Consequently, this techniques suffer from a combination of the problems that SDTA and MBDTA approaches have.

2.1.2 Probabilistic Timing Analysis

Probabilistic methods [45, 32, 42], differently to deterministic ones, provide a Probabilistic Worst-Case Execution Time (pWCET) which materializes in a probability distribution of WCET. The final result of applying PTA is an association of WCET with a probability of exceeding such WCET. This allows us to reason on tighter WCET with an accountable safety compromise. Figure 2.3 illustrates the results from applying different WCET estimation techniques.

- **Static-Probabilistic Timing Analysis (SPTA)** [47]. SPTA uses a discrete spectrum of probabilities of events with associated latencies. Then, for a given task it combines them using proper convolution operations to finally derive a discrete distribution of WCET with attached probabilities.
- **Measurement-Based Probabilistic Timing Analysis (MBPTA)** [34, 48]. MBPTA involves the use of measurements together with statistical analysis methods that allows the end user to derive a probabilistic curve of pWCET. The advantage of this method resides in bringing together the lowered costs of Measurement-Based analysis and the scientific support to the safety margin by using mathematically backed statistical analysis tools. The focus of this work revolves around the MBPTA technique and the randomization solutions that sprew from it.

- **Hybrid-Probabilistic Timing Analysis (HPTA)** [49]. Similarly to the previous hybrid variant, HPTA combine probabilistic measurement-based and static techniques.

2.2 MBPTA Methodology and Requirements

MBPTA [34, 33, 50, 51] is a probabilistic framework to derive probabilistic timing bounds for real-time tasks. MBPTA builds upon two central elements: a platform with specific support to simplify reliable statistical analysis, and a statistical WCET estimation tool. MBPTA requires that execution time measurements during the analysis phase of the system are representative and collected under conditions that reliably upper-bound the system behavior during operation [34, 33]. For that purpose, some components are time-randomized to relieve the end user from having to exercise any low-level control on the platform timing, and instead let randomization expose corner cases.

Time-Randomized Processors (TRP) facilitate the use of MBPTA in complex processor architectures. To use the MBPTA methodology, the platform under analysis must be able to control the sources of time jitter. This can be done in two ways:

- **Time Upper-bounding**. This solution consists in forcing hardware resources to always operate at maximum latency, therefore effectively always providing execution time upper-bound. One example of this would be forcing the floating-point unit to operate at maximum latency independently of its input values.
- **Randomization**. This approach consists in transforming the sources of time jitter into a random information source. For instance, by randomizing cache interference, memory access latency will show a randomized behavior.

Note that MBPTA can be applied regardless of the strategy adopted since both of them provide control over the jittery sources.

In that regard, the purpose of randomization, and therefore TRP, is to improve representativeness of the measurements. For example, if a task's execution time is measured $1,000$ times on a deterministic platform but each execution maintains the same memory layout, memory cache interference and evictions will not be properly accounted for since the layout may change arbitrarily across software integration steps. In contrast, operating in a randomized environment allows us to test and deploy one different layout each execution, therefore increasing the representativeness of the testing campaign [52, 53].

This is of paramount importance since it allows the end user to detach from the need of controlling all the dependent states and relationships between hardware features of a processor. Instead randomization automatically mangles the processor's states for the user isolating the processor from deterministic and pathological combinations of events. For example, in a deterministic architecture, certain cache layouts and access patters could cause programs to exhibit WCET that can be up to 20 times worse than the average execution time [54]. Discovering this specific situation might be an overwhelming task specially with complex architectures and applications.

2.2.1 Statistical Analysis

MBPTA builds upon Extreme Value Theory (EVT) [55] to predict the probabilities of high execution times (probabilistic WCET, pWCET, estimation). EVT is a branch of statistics used to predict extreme (rare) events in fields like finance (major stock market incidences) and hydrology (river floods). EVT tags its outcomes (events) with a probability with which the event is predicted to occur. To that end, EVT models the largest (tail) values measured from the phenomenon under analysis. Based on an execution time sample, an appropriate use of EVT [34, 33, 50] allows delivering reliable exceedance probability bounds for high execution times, including values above those observed.

The key advantage of the use of EVT, as part of MBPTA, is that it is a black-box method that can be applied on any type of sample provided that it complies with certain statistical requirements and has in fact been used in other domains such hydrological, meteorological or financial. However, the obtained distribution is relevant only for the system sampled. In the case of embedded systems, this implies that execution conditions used during analysis match (or upper-bound) those during operation, which is a too demanding constraint in the general case, especially for increasingly complex hardware and software. The use of time-randomization (in the case of timing) allows guaranteeing representativeness of analysis conditions with respect to operation ones, and thus, simplifies obtaining reliable pWCET estimates.

For the Measurement-Based Probabilistic Timing Analysis using Coefficient of Variation (MBPTA-CV) implementation [33] in particular, MBPTA application is a straight-forward process. In the following list we enumerate and explain the necessary steps to obtain a pWCET through MBPTA.

1. **Perform task executions:** First, each task is executed in the MBPTA-compliant platform. A platform can be made MBPTA-compliant by guaranteeing that timing jittery sources are either upper-bounded during analysis time or by forcing a probabilistic behavior (i.e., randomization) on them during analysis and operation [13, 32]. The execution time of the task under analysis is sampled repeatedly. MBPTA-CV imposes the use of a sample sufficiently large so that the number of high values is sufficient for a very tight pWCET estimation (typically between few hundreds and few thousands of measurements). Usually the most common number is around $1,000$ executions. However,this number actually depends on the application and the degree of confidence that must be reached.
2. **Check for statistical representativeness:** One of the most important requirements that MBPTA must comply with, is that data gathered must be extracted from a representative scenario. In order for WCET estimates to hold, validators must enforce that conditions during analysis (while development is going on) upper-bound those at operation (once the system is deployed). This is necessary since the statistical treatment that will later be applied only acts as a black-box without considering specific parameters of the observations taken. One example of topics falling into this concern is program path coverage [56].

MBPTA-CV also tests whether the pWCET can be reliably upper-bounded with an exponential distribution. This occurs for any distribution with a maximum value, even if such value is unknown (i.e., a maximum exists even if we do not know it). This property holds for the distribution sampled, since real-time programs have a finite duration, and so a maximum value in terms of cycles. Hence, the sample of such execution time distribution also meets the statistical property that allows upper-bounding it with an exponential tail.

3. **Check for Independence and Identical Distribution (i.i.d.):** To obtain guarantees that statistical treatment can be used on a set of measurements, those must be first tested for appropriate statistical treatment condition. This means that execution time measurements must have two key properties. In the case of EVT, these are, that the values are independent[1] among them, and also that they are identically distributed[2]. This can be easily tested by using statistical tools like the Kolmogorov-Smirnov test [57] or the Ljung-Box test [58] that assess the independence or the identical distribution properties of a sample. Typically this holds probabilistically for any MBPTA-compliant platform and hence, samples converge statistically to these properties.
4. **Apply Extreme Value Theory (EVT):** Finally, with i.i.d. tests passed, EVT can be applied to model the extreme execution times and therefore provide a pWCET curve. MBPTA-CV delivers a pWCET fitting an EVT distribution with shape parameter $\xi = 0$, thus with exponential slope. The pWCET curve represents the highest probability with which one run of a task exceeds a time bound. Figure 2.4 (right) shows an illustrative pWCET distribution for which the probability of the task exceeding 7 ms is below 10^{-10} per run. MBPTA is well-settled with industrial case studies performed in automotive, avionics, and space [59, 60, 61].

2.3 Processor and Code Architectures favoring Randomization

Current cutting-edge processors are comprised of a myriad of components aimed towards improving average performance. In the context of CRTES, safety and predictability are of more importance than high-performance, hence CRTES processors are simplified versions of their high-performance relatives. For instance, high-performance processors include features that allow to speculatively execute instructions ahead of time, branch prediction or out-of-order pipelines. Instead, CRTES processors contain a much smaller subset of high-performance features due to high-performance designs proving too complex to verify for critical applications. In the

[1] Two events are deemed independent if the occurrence of one does not alter the occurrence probability of the other.

[2] Two events are deemed identically distributed if they can be modeled after the same probability distribution.

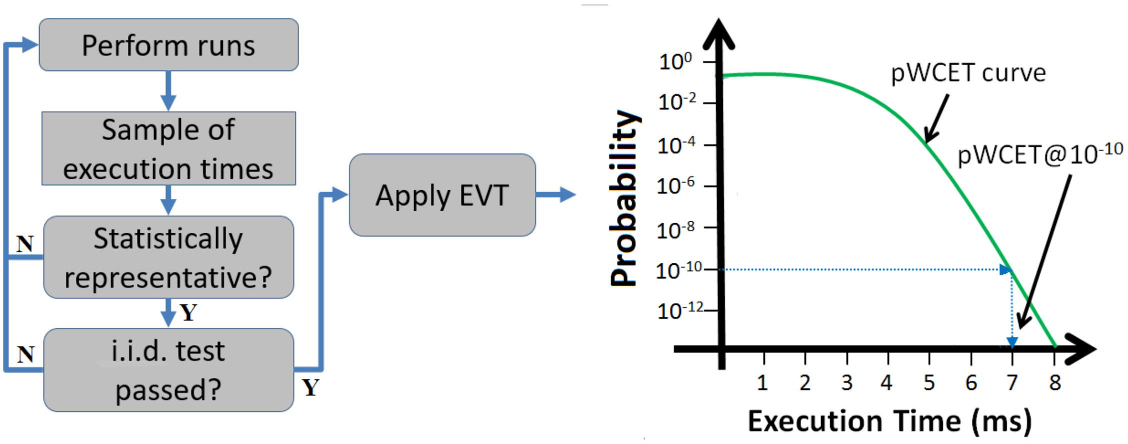


Figure 2.4: MBPTA process and example of pWCET curve. In this example, the task is expected to take more than 7 ms at most once every 10^{10} runs.

context of this thesis (and CRTES in general) we focus mainly on bus connected multi-cores, with in-order execution, no branch prediction or static branch prediction and a low count number of processor stages. Regarding the interconnect component, efforts to analyze the Network-on-Chip (NoC) for CRTES and provide support for many-cores have been made in other studies [62]. In this thesis though, we will focus on single-core and multi-core processors with few cores.

Randomization has been recently applied to provide guaranteed performance without loss of predictability by randomizing the timing behavior of processors [63]. There are two main high-performance hardware resources that have been enabled for its use in CRTES with time-randomization, these are: caches and shared resource arbitration policies [32]. In the case of caches, randomization can be applied either at software or hardware level. For arbitration policies, hardware solutions have tackled arbiters of shared buses, which enables the possibility to obtain WCET of multi-core architectures.

2.3.1 Software Randomization

Software solutions to provide MBPTA compliance have emerged to enable the use of MBPTA with Commercial Off-The-Shelf (COTS) processors. Due to the long development and adoption cycles when designing hardware, software randomization solutions ease in the adoption of MBPTA as timing analysis technique. In that regard two different approaches exists, Dynamic Software Randomization (DSR) [64] and Static Software Randomization [65]. The idea behind software solutions is to generate randomized cache layouts by rearranging and changing the placement of program sections in a random fashion. DSR performs a dynamic allocation of code and data segments at runtime during program initialization. Static Software Randomization (SSR) emerges as a solution to safety standards like ISO-26262 [3] which for example, discourages the use of dynamic objects and pointers. Instead, SSR creates random layouts between different binaries of the same program, hence cache layout randomization is effectively done by executing different binaries rather than during program initialization. Without loss of generality this thesis focuses on the hardware applicability of randomization. We refer the interested reader to [64, 65, 66] for further details on software randomization.

2.3.2 Time-Randomized Caches

In the real-time domain, time-randomized caches were initially proposed to provide probabilistic guarantees [67] that tasks do not suffer pathological cache miss patterns that cause a big impact in performance and are hard to capture during testing [68, 52, 53]. The first proposals on random caches presented non-parametric random caches [69, 70] mainly focused towards attaining better average performance. Parametric random placement [31, 71, 37] soon appeared to provide representativeness [32], increase timing predictability and ease the applicability of MBPTA. These properties of time-randomized caches are attained by the fact that cache conflicts become independent of the actual addresses (memory placement) where code and data are placed unlike traditional placement and replacement cache policies.

- **Cache Random Replacement**: The replacement mechanism decides, within a given cache set, which cache lines stay and which cache lines must be evicted to make room for new recently accessed data. In contrast with other deterministic replacement techniques, random replacement provides independence from previous evictions and an equal probability for data to be evicted from the cache set.
- **Cache Random Placement**: Random placement procures the randomization of data placement across sets. In contrast to random replacement, which works within a given cache set, cache lines are allocated in sets using their memory address so they can later be retrieved when accessed. To reconcile the recovery of data with non-deterministic placement, a random number (random seed) is combined using a hash function with the memory address to randomize the destination set. Hence, every time the random seed is changed a different set is chosen even if the memory layout (addresses) is the same. To maintain cache coherency, every time the random seed changes, a cache flush must be performed as placed data from previous random seeds (synonyms) will remain scattered across other cache sets. In the case this data is modified at some point, future random seeds might make accesses use old cache lines with the outdated values if no flush is performed on a random seed change.

Figure 2.5 illustrates the generic architecture for time-randomized caches. The research in random caches has been successful in proving its feasibility for the real-time domain. Time-randomized cache designs have been evaluated with space case studies [35] and implemented in a Register-Transfer Level (RTL) prototype of a 4-core LEON3 processor [72]. While performance on top of time-randomized caches cannot be regarded as higher or lower than that on top of conventional caches with modulo placement and Least Recently Used (LRU) replacement in the general case [36],

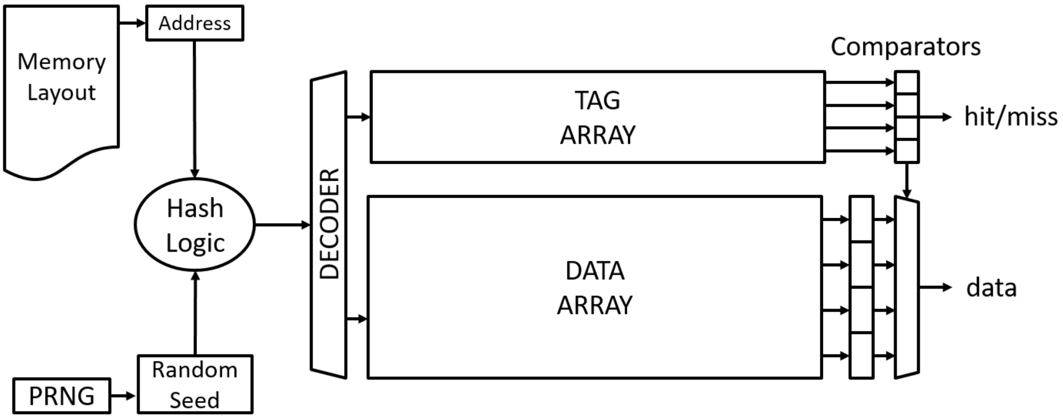


Figure 2.5: Block diagram of a generic time-randomized cache. Address bits are combined with a random seed to access a random set in cache.

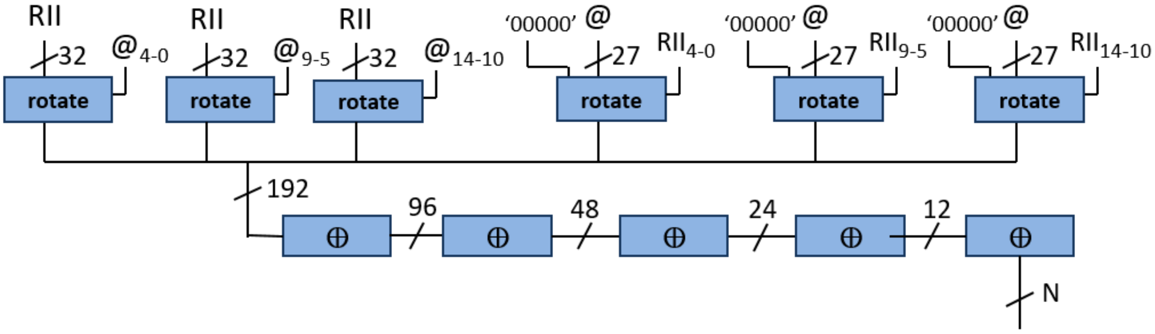


Figure 2.6: Schematic of the hash logic of *hRP*.

they have been shown to provide execution times less than 2% higher than those on conventional caches on average [37]. Note that conventional caches are intended for optimizing average performance while time-randomized ones are intended to provide low pWCET. In this thesis we mainly work with two different implementations of randomized cache placement designs, Hash-Based Random Placement (hRP) and Random Modulo (RM) [37].

2.3.2.1 Hash-Based Random Placement (hRP)

hRP randomly maps memory addresses in the same cache-line boundary to a cache set based on a random seed, which is changed across program runs. *hRP* is implemented using a hash function that rotates address bits based on a random seed (RII) and RII bits based on some address bits. Finally, all bits of those rotations are XORed to obtain the cache set index, as shown in Figure 2.6. This provides independence of the address mapping to sets, hence memory layouts during analysis provide representativeness of memory layouts during operation. While this provides the properties needed by MBPTA, it may produce bad placements in terms of miss rates: even if a program accesses few cache lines, those lines may be randomly placed into the same cache set. Conversely, deterministic modulo (M) placement maps consecutive

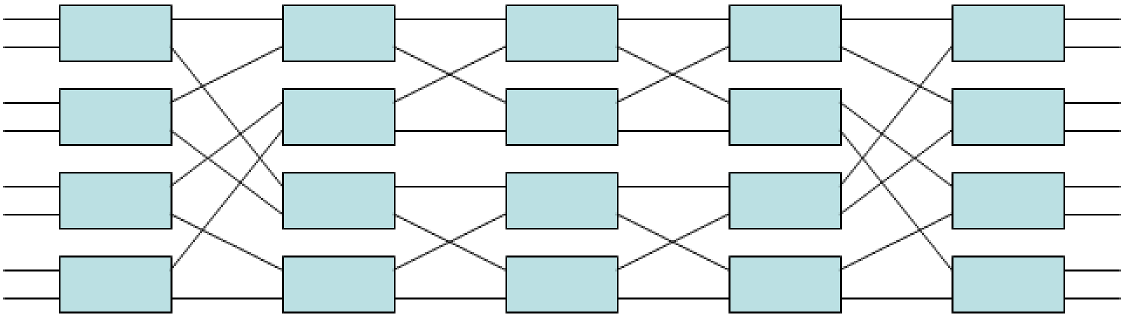


Figure 2.7: Example of a 4-bit Benes network.

memory lines into consecutive sets, thus avoiding this type of conflicts. However, conflicts across lines are not random and strictly depend on memory location. Since memory location of objects during operation is hard to be controlled and mimicked at analysis time, conflicts at analysis are unlikely to represent those during operation, hence thwarting the use of MBPTA.

2.3.2.2 Random Modulo (RM)

RM aims at getting the best of hRP and M worlds. RM randomizes cache placement within cache way boundaries (also using a random seed) so that, as long as cache ways do not exceed memory page size (typical case for L1 caches), consecutive cache lines within cache way boundaries cannot conflict among them by construction, as it is the case for deterministic modulo placement. This is achieved by randomizing placement by randomly permuting the index bits of the memory address therefore taking advantage of data locality (data packed close together with higher chances of being accessed won't contend for the same cache sets). Still, conflicts among lines beyond cache way boundaries are random. Hence, average performance is close to that of deterministic modulo placement and worst-case placements deliver performance close to the average performance [37].

Random modulo is implemented by means of a Benes network where each node allows input signals go through or commute based on control signals (see Figure 2.7). In the case of random modulo, input bits (those coming from the left in the figure) are those address bits used as index in regular modulo placement, and control bits (one control bit per box, not shown in the figure) are produced by XORing conveniently the tag bits[3] and a random seed that is changed across program executions. In this way a given memory line is randomly placed in cache, such placement holds constant during the whole execution, and it changes randomly across executions by changing the random seed (and flushing cache contents). Since addresses within cache way boundaries have distinct index bits, the Benes network delivers a bijective function so that, given specific control signals (those produced by addresses with the same tag

[3] An address includes – from right to left – *offset* bits identifying the bytes accessed within the cache line, *index* bits identifying the cache set accessed, and *tag* bits that identify different addresses placed in the same cache set.

with the same random seed) each index is placed in one set and each set corresponds to exactly one cache index. Therefore, a permutation is obtained and conflicts cannot occur across lines with identical tag.

Note, however, that the particular way to combine tag bits and the random seed determines the particular index bit permutation chosen. Also, the output of the network is determined by the particular index bits of the address being accessed. This is further detailed in Chapter 4, where the reliability assessment is performed and an improved version of the RM indexing is proposed.

2.3.2.3 Hybrid Placement Configurations

Throughout the years, improvements on random cache designs have been proposed to solve the downsides that the first implementations presented. For instance, RM offers substantial improvements over hRP but cannot be implemented at all the cache hierarchy levels. In particular, the RM design implemented in L1 data and instruction caches and the hRP in the L2 cache has been shown to be the most convenient configuration [37]. A priori, *RM* cannot be used in L2 caches since cache ways are much larger than page size, so *hRP* is the only choice for L2 caches. For RM to provide representative measurements the cache way size must be equal to or a divisor of the page size mainly because randomization can only guarantee that conflicts are explored as long as pages map in the same sets. RM explores cache conflicts across pages while avoiding conflicts within the page, if cache ways fit more than one page, the interference between such pages becomes dependent on the memory placement (deterministic) and therefore representativeness during the analysis phase is compromised as memory layouts can change during software integration.

Recently, researchers have implemented randomized placement policies that allow the use of RM in higher-level caches where the way size is greater than the page size. To that end, they propose hybrid placement techniques [73] that randomize the location of pages within the cache way so that interference patterns between different pages are explored.

2.3.2.4 Random Replacement

The initial use of time-randomized caches in real-time systems came from the application of random replacement policies [67]. Under the real-time context, *evict-on-access* [47] and *evict-on-miss* [31] replacement policies emerged as solutions to add timing-predictability to caches. In *evict-on-access* random replacement caches, data can be evicted even when accesses hit in cache. This property allows users to reason on the probability of a value to be evicted regardless of whether memory accesses to a different cache line in the same cache set are hit or miss as the probability of eviction is known. In *evict-on-miss* caches, a cache line is randomly selected for eviction only when the access is a miss. This policy offers greater performance at the cost of predictability. Works like [74] effectively employ this policy for SPTA. Recently, Random Permutation Replacement (RPR) was proposed as a new random replacement policy that improves on previous proposals by avoiding pathological eviction patterns [75].

For the rest of this thesis, we focus on the *evict-on-miss* random replacement policy.

2.3.3 Randomized Arbitration Policies

With the apparition of the power wall in the early 2000s, multi-cores began to popularize as the main route to keep improving performance. However when multiple processing agents are integrated into the same chip, not all elements are replicated, hence creating sharing restrictions for different hardware features. Some examples of these are the buses to shared caches or memory controllers. In order to be able to use them, arbitration policies must be put in place to coordinate access to such resources. However, WCET estimates of tasks that involve the use of shared resources are impacted by the activity of other cores and will need an MBPTA compliant way to account for such activity. One of the most simple ways, which does not require randomization but rather is a form of upper-bounding, is to use Time-Division Multiple Access (TDMA) or Round-Robin to assign a fixed time slot to each of the contenders of a shared resource. Therefore each task can expect to have his own dedicated amount of time for using such shared resource. Although this technique is timing analyzable, since it reliably bounds the amount of time any core can wait for its contenders to liberate the hardware resource, it produces pessimistic WCET due to the need to account for the worst situation possible always, which is to wait for all the contenders to finish with their time slots [76].

On the other hand, randomized arbitration has been shown to offer tighter pWCET and also to outperform the aforementioned deterministic arbitration schemes on average performance. In our case we focus on shared bus arbitration. Basically two different randomized arbitration techniques exist for arbitrating buses in TRP . **Lottery Bus** [77] is the most straight-forward time-randomized implementation were at each arbitration round the access is granted randomly to one of the cores. In this fashion, the probability of not being granted access to the bus follows an exponential distribution that decreases for increasing numbers of arbitration rounds lost, therefore asymptotically approaching a probability equal to 0.0 of not being granted access, and probabilistically bounding how many times an specific core will wait for bus access. **Random Permutations** [76] is an alternative that also has been successfully applied in NoC [78] which in contrast to the Lottery bus, guarantees that probability to be given access to the shared resource will be 1 within a window of access slots. The basic working principle is that the arbiter does not randomly decide on each slot, but rather on a window of slots of size matching the number of contenders, therefore creating random sequences that change on every arbitration round and that will always contain at least once any of the possible cores. Thanks to this scheme pWCET can be further tightened with respect to the previous solutions.

2.4 Non-Functional Metrics

In this section we introduce some background knowledge on the specifics of the different non-functional metrics covered in this thesis. We introduce each metric with emphasis on the aspects relevant for this thesis.

2.4.1 Reliability

Devices that incorporate CRTES are usually expected to showcase higher standards in durability and lifetime mainly because of their costly maintenance and difficult accessibility once deployed. In that sense, reliability of CRTES plays a paramount role as those devices are expected to perform flawlessly due to their criticality for years.

In this reliability context we focus on aging effects, in particular, Hot-Carrier Injection (HCI) [38] and Bias Transistor Instability (BTI) [39]. HCI and BTI are particularly interesting because of their different relation to circuit activity and our purpose will be to study how random placement can mitigate HCI and BTI by balancing the contents of cache cells. In addition to aging considerations, we also make an assessment on how randomized caches and TRP are more resilient to the Resonant Voltage Noise (RVN) phenomenon.

- **HCI Aging:** HCI [38] aging occurs when a carrier (either an electron or a hole) is injected from the conducting channel into the silicon substrate or the gate dioxide, where it stays permanently trapped. Then, whenever the gate is intended to charge or discharge current, further electron-hole pairs need to be made due to the trapped carrier, thus affecting negatively both gate delay and leakage, and potentially making the gate fail its specifications. HCI, among other sources of transistor degradation, affects devices proportionally to the activity produced, which in turn depends on the access distribution across cache sets.
- **BTI Aging:** BTI [39] breaks progressively silicon-hydrogen bonds at the silicon/oxide interface whenever a negative voltage is applied at the gate of P-type Metal-Oxide-Semiconductor Logic (PMOS) transistors Negative-Bias Transistor Instability (NBTI) or a positive voltage for N-type Metal-Oxide-Semiconductor Logic (NMOS) ones Positive-Bias Transistor Instability (PBTI). This creates new carriers affecting gate delay and leakage as in the case of HCI. However, BTI degradation does not relate to switching activity as HCI does but, instead, relates to the amount of time MOS transistors spend in conductive mode. This effect is particularly relevant for cache memories since they are typically implemented with the smallest devices (so the ones that may fail earlier) and conventional 6T and 8T cells consist of 2 inverters arranged in a ring fashion so that the PMOS transistor of one inverter degrades due to NBTI and the NMOS transistor of the other inverter degrades due to PBTI regardless of the cell contents. In this context, it has been observed that aging is maximized when the cell stores always the same value, so two transistors degrade constantly whereas the other two do not degrade at all. Conversely, aging is minimized when the cell stores a ’0’ 50% of the time and a ’1’ also 50% of the time [79].
- **Resonant Voltage Noise:** Pathological behavioral patterns can occur in the time dimension if events such as, for instance, memory accesses, occur with

precise frequencies. Using deterministic caches, and arbiters in interconnects and memory controllers, can create those systematic and pathological patterns. A side effect of that occurring is that power dissipation follows those patterns. The synchronization of power demanding events and the frequency at which those events occur has been shown to be the factors with major contribution to the voltage noise in the power distribution networks of multi-core processors [80]. RVN is created by power fluctuations and this effect is amplified when such fluctuations are repetitively caused by the synchronization of high power consuming events. This may cause severe voltage droops and hence, failures affecting all tasks running in the processor.

Thesis focus: The items previously described downgrade lifetime and safe operation of CRTES. The goal of this thesis (and Chapter 4 in particular) regarding these reliability hampering effects is to leverage how TRP mitigate them and propose designs that further improve the lifetime of CRTES and their reliable operation while preserving MBPTA compliance for pWCET estimation.

2.4.2 Security

Connected vehicles have become one of the major goals of car makers given their potential to, for instance, provide the user with new software updates that add new features and enhance existing functionality [27]. These interconnection capabilities compounded with the utilization of high-performance processor features, can be used by malicious software to perform several types of attacks. In particular, we mainly focus on Cache-Timing Side-Channel Attacks (SCA) [81] but also consider Unauthorized Control Information Tampering (UCIT) [82] and Denial of Service (DoS) [83].

2.4.2.1 Unauthorized Control Information Tampering (UCIT)

UCIT vulnerabilities include many common software-related security problems such as buffer overflow, format string, integer overflow, and double-freeing of heap buffer [84, 85]. Attackers exploit these vulnerabilities by changing control information (e.g., processor state registers) with the purpose of pointing to the attacker's malicious code. These attacks do not depend on the potential actions on the user side but simply exploit existing program bugs to attack the system.

2.4.2.2 Denial of Service (DoS)

In high-performance multi-core processors, some key resources are shared among running tasks. Resource sharing allows processors to improve area and power efficiency but introduce a side effect on the security, and also time-analyzability, when processors do not provide sufficient performance isolation properties. Multi-core processors are vulnerable to DoS attacks since one shared resource, typically the memory system, can be unfairly shared among multiple cores. In this context, a malicious task can compromise the performance of another task running in the same processor by clogging a shared resource, significantly affecting the performance of co-running tasks or even precluding resource utilization by others. Intuitively, one may think that this effect only arises in processor designs for the mainstream market with limited performance isolation properties, however, this effect has also been observed in some processors targeting the real-time domain [86].

2.4.2.3 Cache Timing Side-Channel Attacks (SCA)

Side-channel attacks exploit system's information leakage to the physical environment when an encryption or security process executes in order to steal cryptographic keys or interfere with information processes. With *Cache Timing-Based Side-Channel Attacks*, which we will simply refer to as SCA, the attacker infers information about the keys based on the execution time variability caused by cache memories [87, 41, 88]. SCA exploit the difference in time that memory access patterns expose; in particular, hit and miss patterns that occur in caches. When these patterns are related to the placement of the data in memory, for instance, attackers can exploit the deterministic behavior of high-performance computing caches to extract cryptographic keys [41].

In particular, SCA are enabled by two basic principles:

1. The time difference between accesses: misses on a cache take longer to resolve than hits, hence leaking which data is being used and present in cache, and which data is not in there.
2. The use of lookup tables that are input dependent in cryptographic algorithms (e.g., Advanced Encryption Standard (AES)).

Following those principles, two different types of attacks can be performed on caches:

Contention-based attacks. In this thesis we focus on the particular SCA attack referred to as contention-based attacks [89]. In this case an attacker contends for the same cache sets with the victim process, potentially leading to eviction of one's cache line by the other. When the contention and eviction is deterministic, the attacker can infer the memory address (determined by the value accessed) of the victim based on the cache sets that have been accessed.

Reuse-based attacks. Reuse attacks [89] exploit the shorter execution times experienced by memory accesses when fetched data blocks are stored in the cache (i.e they are reused). Victim or attacker processes will execute faster if one the opposing tasks has brought sensitive data into the cache, therefore leaking such information.

Thesis focus: Although security is nowadays one of the main concerns in computer architecture, most of the time it is not addressed until major flaws and attacks are discovered. In this thesis (specifically in Chapter 5) we aim to provide security for MBPTA-compliant CRTES against SCA and other attacks.

2.4.3 Energy

The proliferation of battery-powered devices together with the power wall and the drive towards power-efficient systems have made energy a first-class citizen for CRTES. Nowadays, a lot of system design decisions are influenced by energy consumption. Many CRTES usually have to operate under limited energy availability (e.g., satellites and drones) and must ensure that they execute critical tasks to completion. Because of these reasons, energy accountability and validation is of paramount importance, specially so, when critical applications are involved. Hence, CRTES require thorough energy validation and accurate energy models.

In this thesis we tackle two energy related topics, worst-case power/energy estimation and peak power validation. To understand the intrinsic behavior of energy consumption, it is necessary to comprehend how energy is usually accounted for. To that end, we first start by introducing the basic concepts of power estimation with a well accepted general energy consumption model of processors and its parameters. Throughout the energy related chapters we mainly build on power formulation (rather than energy), although power and energy can be used interchangeably given a fixed execution time t. The relation between power (P) and energy (E) is given by $E = P \cdot t$.

Total dissipated power can be classified into two complementary terms as shown in equation 2.1.

$$P_{total} = P_{stat} + P_{dyn} \tag{2.1}$$

Static power (P_{stat}) dissipates when maintaining a circuit powered up. It covers the power dissipated through leakages, free carriers (electrons and holes) that are able to escape the isolation layers of the silicon. There are several models for deriving static energy consumption but a widely accepted formulation is shown in Equation 2.2.V_{cc} is the nominal voltage for the circuit; N the number of transistors; k_{design} an implementation dependent constant; and $I_{leakage}$ the leakage current that depends on the technology used for the chip implementation [90].

$$P_{static} = V_{cc} \cdot N \cdot k_{design} \cdot I_{leakage} \tag{2.2}$$

Interestingly, N and k_{design} are truly constant parameters, while V_{cc} and $I_{leakage}$ are theoretically assumed constant, but they can actually suffer some fluctuation. V_{cc} may vary due to techniques for power saving such as Dynamic Voltage and Frequency Scaling (DVFS) and drowsy operation modes [91]. V_{cc} also depends on the quality of the voltage supply source and the chip package. It further suffers from significant fluctuations at operation time, especially in multi-core setups [80], since the likelihood of abrupt power dissipation variations increases due to, for instance, several cores having high energy consumption requirements at the same time. This creates current glitches and thus, voltage droops. $I_{leakage}$ highly depends on the thermal status, so that high temperatures increase the leakage current, thus increasing the dissipated static power.

It is also worth noting that although k_{design} is constant, it is an approximation to abstract the internal complexities of processor designs and also depends on the individual chip fabricated since process variations lead to variations across chip units.

Despite those sources of variation, P_{stat} is often assumed constant due to it being highly stable over time, which makes nominal P_{stat} estimates be very precise with respect to average behavior. However, this does not necessarily hold for maximum P_{stat} estimates, which are the ones of interest in our proposals.

Dynamic power (P_{dyn}) dissipates due to the charging and discharging of transistor's gate capacitance and can be expressed as shown in Equation (2.3), where A is the switching activity or activity factor, representing the percentage of transistors' capacitance flipping value, V_{cc} is the nominal voltage, C_{eq} is the equivalent capacitance of the transistor inputs and f is the operating frequency of the device.

$$P_{dyn} = A \cdot V_{cc}^2 \cdot C_{eq} \cdot f \tag{2.3}$$

In general, all those parameters are subject to variations, and so it is P_{dyn}.

A strongly depends on the input changes of the components. Those inputs include data and control signals of the circuit. As an example, A for an adder depends on the input data variation as well as on the control signals to add/subtract, etc. Deriving approximations to A has been the subject of intense research [92]. It has been observed that A decreases exponentially across gate levels when moving from inputs to outputs [92]. However, this cannot be proven in general and the exponential factor can only be approximated for specific circuit types. Thus, to the best of our knowledge, reliable and tight upper-bounds to the activity factor usable for any type of circuit do not exist.

V_{cc} suffers from the same variation effects explained before. In the case of DVFS, both V_{cc} and f vary coordinately. In that case, we regard f as constant with respect to V_{cc}, so that given a nominal V_{cc} value, a given nominal f is set. In practice, f may change when the clock source is subject to some form of variation, such as, for instance, temperature variations, which may slow down or speed up the clock slightly given a fixed V_{cc} value. C_{eq} is a nominal value that depends on the size of the transistors and it is also subject to process variations introduced during manufacturing.

Thesis focus: Parametric models have been widely employed to predict energy consumption of processors. New CRTES will increasingly be made energy-aware and their attached criticality demands more rigorous and accurate predictions. Accuracy of models is challenged when faced by increasing complexity and parametric variability. In this thesis (particularly Chapters 6 to 8) we address how current models like the one previously described show limitations regarding CRTES necessity of rigorous estimations and we propose suitable methods to leverage uncertainty in the energy domain building upon MBPTA-compliant processor designs and selected methodologies.

Chapter 3

Experimental Setup

> *"A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it."*
>
> — Albert Einstein

The experimental setup used to evaluate the proposals made in this thesis covers four main aspects: Modeled processor architectures, tools and simulators, benchmarks, and other methodological considerations.

3.1 Architectures

We mainly model three different architectures from the Critical-Real Time Embedded Systems (CRTES) domain: the Cobham Gaisler's LEON4 [93], the Cobham Gaisler's NGMP [94] and the NXP e200z4 [95]. Although their architecture is similar some small differences arise between them that made them more suitable for particular experiments. For instance, the Cobham Gaisler's LEON4 has been extensively used in space applications and hence it's a perfect representative of that domain. The NGMP is the natural evolution of the LEON4, as it includes a multi-core system that permits to evaluate multitask workloads. Finally, the NXP e200z4 is a well-known automotive processor, representative of automotive applications. By the time this thesis started many-core and General Purpose Graphics Processing Units (GPGPUs) made small impact in CRTES architecture, therefore in this thesis we focus on simple in-order bus-connected multi-cores for which code and memory performance is the key and whose deployment in commercial systems is still in process.

In Table 3.1 the microarchitectural details and options of each architectures are detailed. Figure 3.1 displays one of our reference architectures, the Next Generation Microprocessor (NGMP), which also shows the internal architecture of the LEON4 since the NGMP is composed of LEON4 cores.

- **LEON4 Core.** The LEON4 [93] is the successor of the LEON3 [96] processor and our reference processor architecture on which we base most of our assessments. The LEON4 is an in-order core that comprises a pipeline with 7 stages:

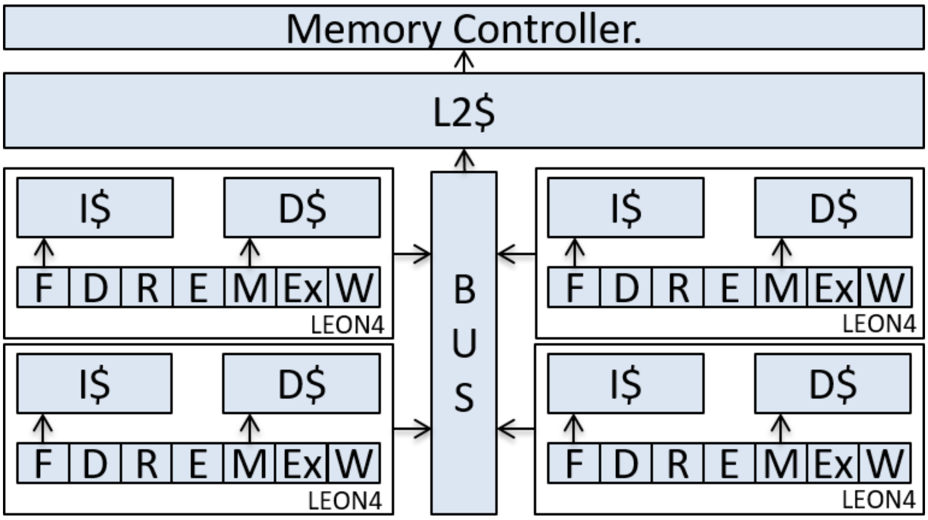


Figure 3.1: Simplified block diagram of the NGMP Architecture. Our modeled reference LEON4 architecture would contain only one core while the simplified NXP e200z4 would substitute the LEON4 for the NXP e200z4 architecture.

Fetch, Decode, Register Access, Execute, Memory, Exception and Write-Back implementing a SPARC V8 [97] Instruction Set Architecture (ISA). It features configurable instruction and data L1 caches and a unified L2 cache together with write buffers and a fully pipelined IEEE-754 Floating-Point Unit (FPU). Although the LEON4 code name refers to the architecture of a core, throughout this thesis we use indistinctly the LEON4 term to refer to the implementation of an entire single core LEON4 processor including cache memory hierarchy and memory controller.

- **NGMP Multi-core.** The NGMP [94] is the European Space Agency (ESA) processor for the future space missions which we model as our reference multi-core architecture, in particular the GR740 implementation [98]. This processor is composed of 4 LEON4 cores with an AHB AMBA bus interconnect, shared L2 cache, and memory controllers. Floating-point units have two separate datapaths, one for square roots and divisions and another fully-pipelined one for the rest of the floating-point operations.
- **NXP e200z4.** We also emulate a processor resembling the NXP e200z4 [95] a single-core automotive microcontroller chip that implements a POWER ISA architecture. We model a 5-stage in-order processor with Instruction Fetch, Decode, Execution0, Execution1 and Result feed-forward stages. We model this architecture with a complete cache hierarchy, separate instruction and data L1 caches (4-way, 128 sets) and unified L2 cache (4-way, 2048 sets). To better comprehend the behavior of the security challenge, instruction prefetching is disabled and a single-issue implemented instead.

Parameters		**LEON4**	**NGMP**	**NXP e200z4**
Core	Stages	7	7	5
	Core Count	1	4	1
	Write Buffer Size	2		1
	Clock Rate	700 MHz		
	Operating Voltage	0.9 V		
L1 I/D Cache	Line Size	32 Bytes		
	Way Size	128 Sets		
	Associativity	4 Way		
	Total Size	16 KiB		
	Write Policy	Write-Through		
	Placement Policy	M, hRP, RM		
	Replacement Policy	LRU, Random Replacement		
L2 Cache	Line Size	32 Bytes		
	Way Size	2048 Sets		
	Associativity	4 Way		
	Total Size	256 KiB		
	Write Policy	Copy-Back		
	Placement Policy	M, hRP		
	Replacement Policy	LRU, Random Replacement		
Bus	Arbitration Policies	Round-Robin, Random Permutations		
	Width	128 bits		

Table 3.1: Summary of modeled reference architectures. LEON4 as a single core architecture, the NGMP as the multi-core reference containing 4xLEON4 and the NXP e200z4 as a simpler reference automotive architecture.

3.2 Tools & Simulators

We build on simulations in order to evaluate and test the hypotheses in this thesis. For modeling microarchitectural timing and functional behavior we use a modified version of SoCLiB [99] simulator. SoCLiB is an open virtual prototyping tool for multiprocessors and a SystemC based simulator. This tool allows us to perform Cycle-Accurate Bit-Accurate (CABA) simulations of System-on-Chip (SoC) architectures. We configure the SoCLiB simulator to emulate the different embedded architectures. Accuracy of the timing simulator has been assessed to be on average 3% off from the reference N2X implementation of the NGMP [100]. SoCLiB's improved simulator builds on a decoupled two-fold architecture were functional emulation and timing simulation are separated which brings flexibility by allowing the implementation of different ISA that can use the same microarchitectural timing model.

We improve our microarchitectural simulator by attaching the McPAT [101] power simulator. McPAT is the MultiCore integrated Power, Area and Timing Modeling framework from HP labs that provides estimates on energy, timing and area by using architectural descriptions of the hardware and parametric performance values. By attaching both simulators together we can obtain either instantaneous or global precise energy and power estimations of our platforms. Additionally, for one of the contributions we also involve the use of the McPAT-PVT. McPAT-PVT is an extended version of the McPAT simulator that not only offers energy and area estimates but also reports how those values vary under the presence of Process Variation (PV). McPAT requires two different types of inputs: Parameters that define the architectural model (e.g., number of L1 caches and number of cores) and parameters that define the simulation (e.g., cycle count and instructions executed). The architectural parameters are statically defined according to each of the reference architectures and the simulation parameters are extracted from the performance metrics of SoCLiB. This pipelined behavior is illustrated in Figure 3.2.

In some contributions we include information about area and delays from hardware designs. When evaluating hardware designs' implementations we use as synthesis tool the Synopsys Design Compiler [102] and make use of the TSMC 45 nm technology library [103] before place & route. In the cases where we resort to hardware simulation we use the QuestaSim [104] simulation framework instead. Note that we use different technologies for power simulation (McPAT) and hardware implementations, the reasons behind this lay on the higher relative accuracy of McPAT for power measurements.

Finally we make use of the MBPTA-CV tool that generates the pWCET estimates and curves and checks of i.i.d properties. This tool is based on the application of MBPTA and EVT using the Coefficient of Variation (CV) algorithm [33]. MBPTA-CV is implemented using R statistical software and programming language [105].

3.3 Benchmarks & Applications

We mainly use benchmarks from one source, the EEMBC Automotive suite [106] and complement them with 3 other sources of benchmarks: custom designed micro kernels, two space use-case applications from the European Space Agency [107, 108], and a couple of Mälardalen benchmarks [109] that add variety to our evaluation. The complete benchmark suite used along with the brief description can be seen in Table 3.2. Benchmarks are executed without Real-Time Operating System (RTOS) support for simplicity of implementation and to avoid interference due to preemption or other RTOS mechanisms.

- **EEMBC Automotive.** The Embedded Microprocessor Benchmark Consortium (EEMBC) [106] Automotive suite is a compilation of benchmarks that perform control of automotive functions such as gear rotation, ignition systems, and more. The suite is purposefully designed to provide measurements of performance in embedded processors. The inner workings of the EEMBC are very simple, the binaries embed input data that would come from sensors, and a main body loop calls different functions that actuate on such data. The EEMBC allow configuration of the number of iterations the main body loop must perform.
- **Mälardalen.** The Mälardalen open-source benchmark [109] suite is also another well known suite for real-time systems. In contrast to the EEMBC, the Mälardalen benchmarks contain a simpler version of program structure that mainly consists of small loops and linear code. This is because the Mälardalen suite was originally intended to be used for WCET tool analysis. In this thesis we make use of the matmult and firFn benchmarks as probes for mathematical linear equation computation and nested loops respectively.
- **ESA benchmarks.** In the case of the ESA benchmarks we make use of DEBIE [107] and OBDP [108] benchmarks. The DEBris in orbit Evaluator (DEBIE) manages an instrument for small space debris and micrometeoroids observation, by detecting their impacts on its sensors. DEBIE has been part of PROBA-1 satellite. The On-Board Data Processing (OBDP), is part of the near infrared (NIR) HAWAII-2RG detector. This algorithm processes raw frames provided by such detector.
- **Microkernels.** Finally, and for specific experiments, we have developed custom microkernels to stress particular regions or behaviors of the processor. Our main objective with these benchmarks is to generate controlled synchronization or desynchronization of events to observe power surges and pathological events. These microkernels are basically composed of instructions that stress as many regions of the processor as possible and then contain a mix of long latency instructions its number depending in case we wanted synchronization or not. To that end, we generate two different microkernels, one that synchronizes high power demanding events, and another one that does not synchronize events.

EEMBC Autobench	
a2time	Angle to Time Conversion
aifftr	Fast Fourier Transform (FFT)
aifirf	Finite Impulse Response (FIR) Filter
aiifft	Inverse Fast Fourier Transform (iFFT)
basefp	Basic Integer and Floating-Point
bitmnp	Bit manipulation
cacheb	Cache buster
canrdr	CAN Remote Data Request
idctrn	Inverse Discrete Cosine Transform (iDCT)
iirflt	Infinite Impulse Response (IIR) Filter
matrix	Matrix Arithmetic
pntrch	Pointer Chasing
puwmod	Pulse Width Modulation (PWM)
rspeed	Road Speed Calculation
tblook	Table Lookup and Interpolation
ttsprk	Tooth to Spark
Mälardalen benchmarks	
matmult	Multiplication of two 20x20 matrices
firFn	Finite Impulse Response (FIR) Filter 700 items sample size
ESA Applications	
OBDP	On-Board Data Processing
DEBIE	Debris inOrbit Evaluator
Microkernels	
pvsync	Power virus with synchronization
pvunsync	Power virus without synchronization

Table 3.2: Summary of benchmarks used for evaluation.

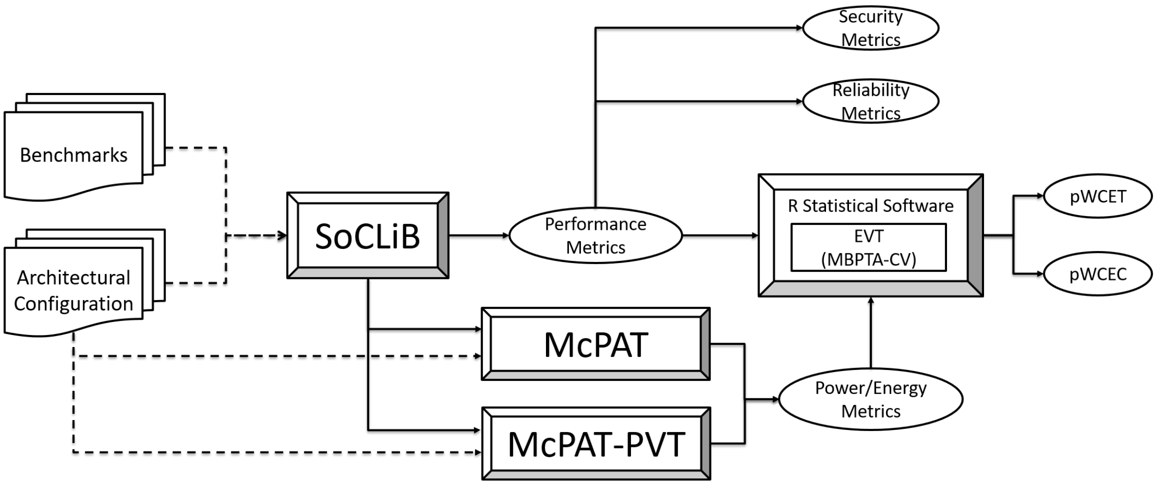


Figure 3.2: Overview of the evaluation framework. Benchmark's binaries and parametric files are inputs to our microarchitectural simulator, that provides performance metrics. This performance metrics can later be processed by the energy simulator (McPAT), used to extract conclusions, or passed to the MBPTA-CV algorithm implemented in R to perform statistical analysis and derive probabilistic estimates.

3.4 Methodology

The metrics that we obtain from our experimental framework are shown in Figure 3.2 inside the elliptic shapes. For reliability evaluations, we rely mainly on performance metrics obtained from our microarchitectural simulator, SoCLiB. Measurements to prove our hypotheses in the security assessment will also come from SoCLiB's available performance metrics.

In the case of energy we feed the performance metrics to the power and energy simulators McPAT and McPAT-PVT, which provide us with power and energy estimates. Finally, to compute probabilistic estimates and project tails of measured distributions we apply Extreme Value Theory (EVT) using the Measurement-Based Probabilistic Timing Analysis using Coefficient of Variation (MBPTA-CV) algorithm from which we obtain probabilistic Worst-Case Execution Time (WCET) and Worst-Case Energy Consumption (WCEC) in the energy evaluation. The MBPTA-CV implementation uses the open source R statistical software tool to compute such estimates.

For multi-core executions we create two types of workloads. The first type involves a single task under analysis that will be executed until completion while contenders run on the background (all other tasks iterate indefinitely on the background). The second type is for a multitask unit of analysis (simulation will be stopped once all the tasks have at least completed once). Which methodological process is followed is detailed in the evaluation section of each of the chapters whenever multi-cores are involved.

For energy modeling we build on McPAT. McPAT computes energy estimates from event counters that can be provided either at any given instant or for the complete execution of a task. Each of these events activates certain processor features, wires

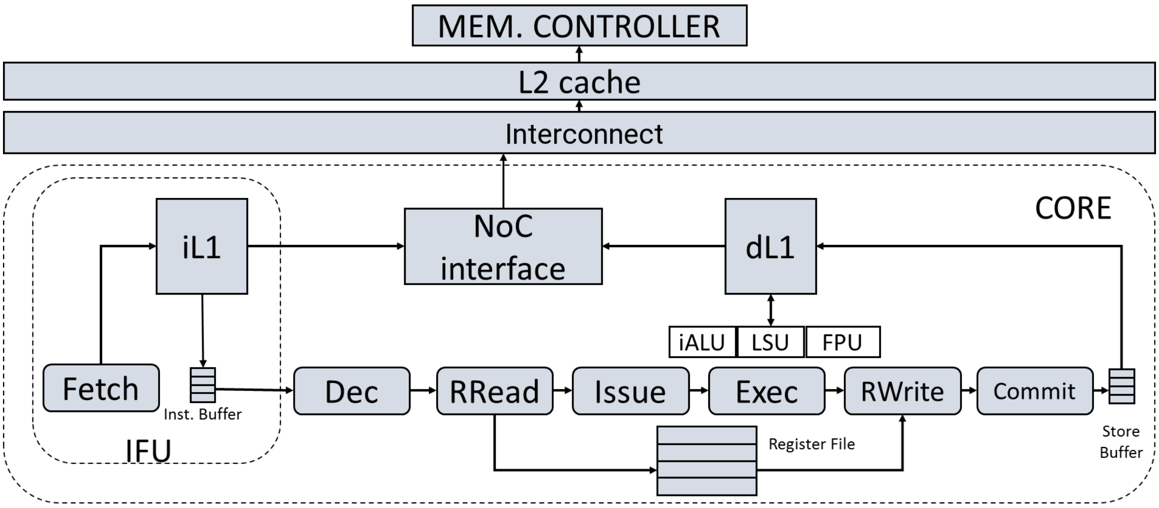


Figure 3.3: McPAT functional unit blocks nomenclature mapped to a detailed NGMP/LEON4 Architecture. The only changes with respect to SoCLiB's model are the naming convention, both simulators allow the emulation of a 7-stage pipeline.

or cells that have an associated cost in energy. This energy cost is then leveraged by McPAT together with the usage and access counts to compute the energy estimate. The addition of all the energy costs of all these events makes for the complete energy estimate that given an execution time can be then turned into dissipated power. Regarding the parameters, unless indicated otherwise, architectures are modeled after 90 nm fabrication process[1] and set to 0.9 V nominal voltage. In Figure 3.3 we can see all the high level components modeled by McPAT, which we refer to as Functional Unit Block (FUB). As it can be seen, the configured McPAT model maps very well to the NGMP or LEON4 architecture (depending on whether multi-core or single-core is used). The main features and FUBs include: instruction fetch unit (IFU), load-store Unit (LSU), register file (RegF), integer ALU (IALU), floating-point unit (FPU), result broadcast bus (RBB), L2, NoC, and memory controller (MC).

Due to the nature of event counting in our performance simulator, to compute meaningful energy estimates, we must first ensure that we measure a time window that at least reports a new event. Our longest latency operation that would stall a single core architecture for the longest of time takes around 30 cycles, therefore the granularity at which we can measure instant energy consumption with our performance and power simulators will be of 43 ns, at a standard frequency of 700 MHz. Summarizing, for instantaneous energy estimates granularity is of 43 ns, unless stated otherwise. Finer sample rates have not been considered due to the intrinsic limits of the power model, which fails to spread power dissipation of an event across multiple cycles, thus creating anomalies at too fine rates (e.g., every cycle). For global energy estimates, granularity will be the full task duration.

[1] As previously stated, simulation technology differs w.r.t. synthesis due to exhibiting higher relative accuracy.

Part II

Reliability in CRTES

Chapter 4

Enhanced Randomized Cache Designs for Improved Aging and Reliability

> *"In life, unlike chess, the game continues after checkmate."*
>
> — Isaac Asimov

4.1 Introduction

Systems such as those in the avionics and automotive domains (among others) deal with functionalities with human in the loop or that relate to the integrity of the system itself. This requires assessing that those systems will perform their operation *correctly* and *in time* following the guidelines in the corresponding safety standards (i.e., ISO-26262 in automotive [3] and DO-178C in avionics [4]).

Measurement-Based Probabilistic Timing Analysis (MBPTA), as stated in Chapter 2 [34], has been proposed to derive reliable Worst-Case Execution Time (WCET) estimates on top of complex hardware. MBPTA benefits from hardware platforms providing some properties such as random placement and replacement in caches [31, 110, 37]. In this line, different implementations of random placement have shown to be suitable for first-level (L1) and second-level (L2) caches. In particular, *Hash-Based Random Placement (hRP)* [110] has been shown convenient for L2 caches whereas Random Modulo (*RM*) has been shown more convenient for L1 caches [37]. While they simplify the use of MBPTA to obtain Probabilistic Worst-Case Execution Time (pWCET) estimates, their fault tolerance (needed for functional correctness) has been barely considered. In fact, the robustness properties of randomized hardware platforms have only been exploited to derive pWCET estimates that hold valid in the presence of faults in random placement and replacement caches [111, 23].

However, the intrinsic robustness of randomized hardware designs has not been assessed yet. As explained before, the number and type of sources of failure due to aging is abundant. The purpose of this work is not being exhaustive in the analysis of all of those sources, but illustrating how randomized hardware designs can contribute positively to mitigate aging at least for some sources of aging.

In this chapter we perform, for the first time, an assessment of the aging-robustness of random placement cache designs: random modulo and hash-based random placement. We propose a new random modulo implementation preserving its key benefits in terms of low critical path impact, low miss rates and MBPTA compliance; while reducing hot-carrier injection aging by achieving a better (yet random) activity distribution across cache sets. On the other hand we show that gains in terms of Bias Transistor Instability (BTI) aging are limited for random placement designs on their own. Additionally, we also introduce some considerations on Time-Randomized Processors (TRP) resilience against Resonant Voltage Noise (RVN). We refer the reader to Section 2.4.1 for a detailed description of these effects.

4.2 Enhanced Aging-Friendly Random Cache Placement

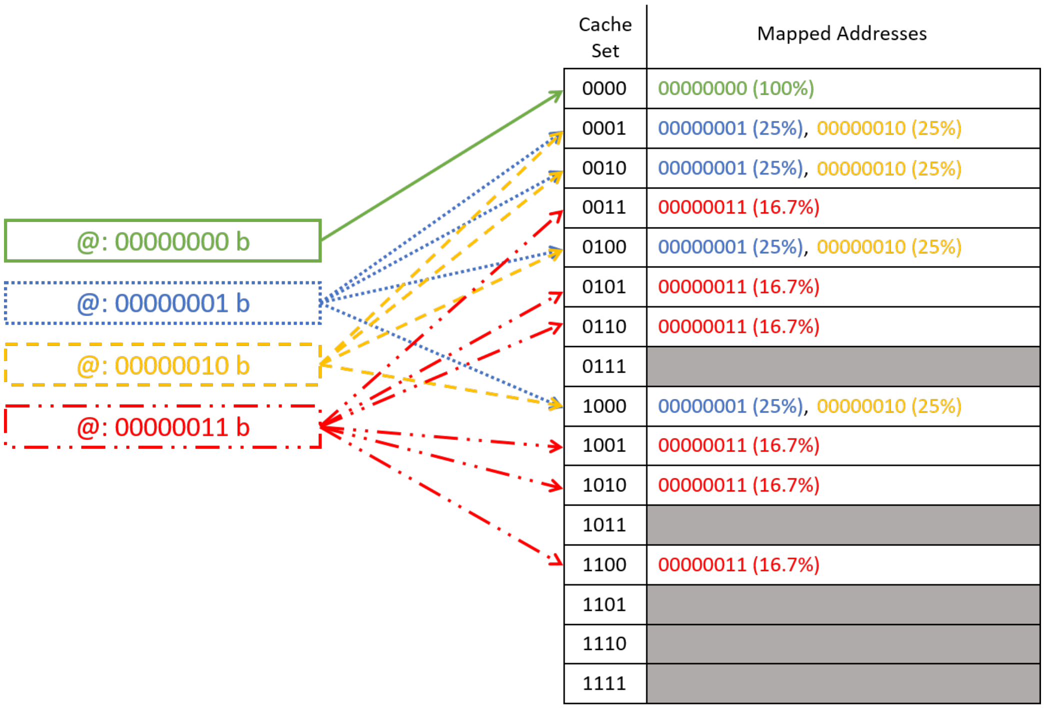
In this section we first describe the functioning of the default *Random Modulo (RM)* cache design, used for L1 caches, in terms of its limitations related to aging. We then present our new enhanced *RM* implementation that makes a far more balanced use of the cache sets, thus more friendly from an (mostly Hot-Carrier Injection (HCI)) aging perspective. Finally, we study *hRP* set distribution for L2 caches, and show that it uses cache sets in a fairly balanced way. In particular we show how the particular address-to-set mapping influences the utilization of each set and, therefore, their degradation in terms of HCI. Hence, we aim at finding a better random modulo implementation that balances utilization of the cache sets regardless of the particular access pattern and indexes of the addresses accessed.

4.2.1 Random Modulo Set Distribution

RM is intended to randomly distribute addresses (indexes in particular) among cache sets. To that end, RM combines address tags with random bits from the random seed to set control signals in the Benes network (see Chapter 2). The goal is to reach a homogeneous and random permutation selection for index bits. However, index bits themselves are not random since they are completely program dependent.

Let us illustrate this with an example. Let us assume a program accessing 4 addresses (once offset bits have been removed): 0x00 (00000000b), 0x01 (00000001b), 0x02 (00000010b) and 0x03 (00000011b), and a cache memory with 16 sets, so that only the 4 lowermost bits are used for choosing the set being accessed. Note that offset bits indicate the bytes accessed within the cache line, but do not relate to cache line identification, so they are irrelevant for this example.

- Interestingly, regardless of the particular bit permutation selected, address 0x00 can only be placed in set 0 (0000b). Since the 4 lowermost bits are “0”, any permutation of them leads to exactly the same set identifier: 0000b.



Cache Set	Mapped Addresses
0000	00000000 (100%)
0001	00000001 (25%), 00000010 (25%)
0010	00000001 (25%), 00000010 (25%)
0011	00000011 (16.7%)
0100	00000001 (25%), 00000010 (25%)
0101	00000011 (16.7%)
0110	00000011 (16.7%)
0111	
1000	00000001 (25%), 00000010 (25%)
1001	00000011 (16.7%)
1010	00000011 (16.7%)
1011	
1100	00000011 (16.7%)
1101	
1110	
1111	

Figure 4.1: Example address distribution for a 16-set cache with default random modulo design.

- Instead, addresses 0x01 and 0x02 can be mapped to any set such that its set identifier contains exactly one "1" given that their binary representation has exactly one "1" and three "0" (0001b and 0010b respectively) in the 4 lowermost bit positions. This corresponds to sets 1 (0001b), 2 (0010b), 4 (0100b) and 8 (1000b). Any other set cannot be reached with bit permutations of addresses 0x01 and 0x02 regardless of the permutation selected since their lowermost 4 bits – those forming the cache index – only contain one "1".
- Finally, address 0x03 can only be mapped to sets 3 (0011b), 5 (0101b), 6 (0110b), 9 (1001b), 10 (1010b) and 12 (1100b) since they are the only ones whose set identifiers contain exactly 2 bits set to "1" (these are the only permutations possible with 0011b as input).

This is graphically illustrated in Figure 4.1, where we can see that the frequency with which each set is used is highly heterogeneous. For instance, set 0 is intensively used by address 0x00 (100% of the times address 0x00 is placed in set 0). Some other sets are used with different degrees of intensity by addresses 0x01, 0x02 and 0x03. For instance, addresses 0x01 and 0x02 are mapped to set 1 25% of the times each (permutation 0001b). Some other sets (7, 11, 13, 14, 15) are never used since no address has enough bits set to "1" to be mapped to any of those sets under any permutation. For instance, since no address has 4 bits set to "1", no bit permutation can make any address access set 15. Further note that, if addresses are accessed heterogeneously, the impairment in the use of the different cache sets can be potentially much higher.

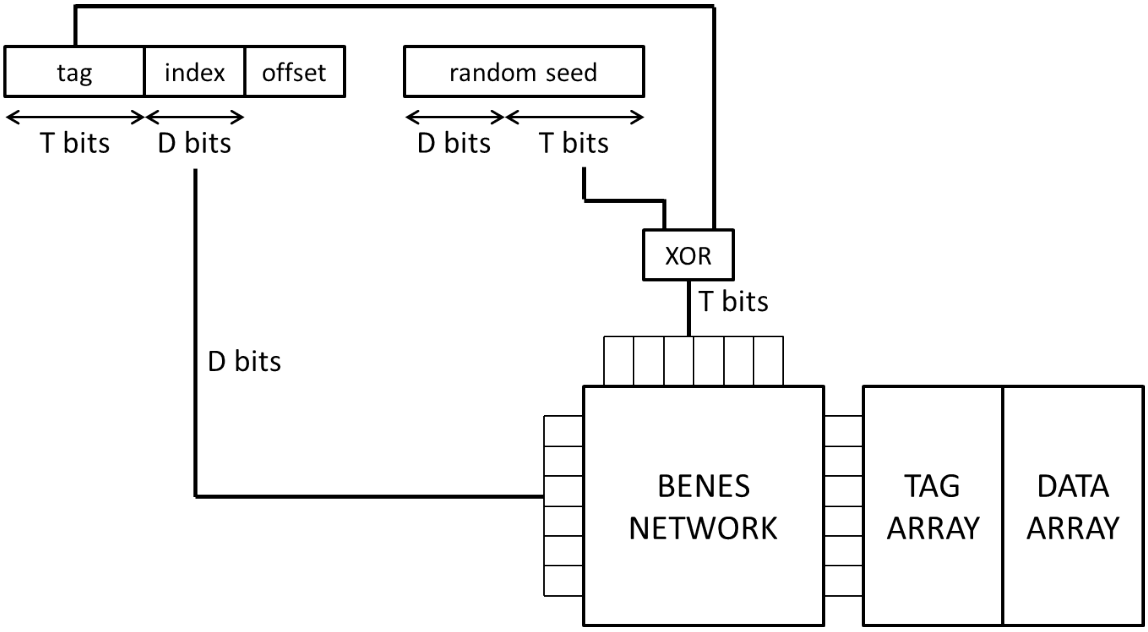


Figure 4.2: Schematic of the baseline implementation of a random modulo cache.

Having an heterogeneous cache set utilization is expected to lead to higher degradation for the most used sets due to, for instance, HCI among other sources of transistor degradation, since HCI affects devices proportionally to their switching activity, which in turn depends on the access distribution across cache sets.

In terms of balancing contents stored, relevant for BTI, placement cannot vary what contents are stored in each particular bit of the line. Thus, if a particular bit (e.g., bit 30) is highly biased towards '0', placement functions cannot change this fact. However, when all addresses make an homogeneous use of the sets, the maximum bias of any particular bit in each position is minimized. For instance, if 90% of the lines have bit 30 set to '0', a deterministic placement may make that some specific sets have bit 30 100% of the time set to '0', whereas others have it set to '0' less than 90% of the time. This would cause the fastest degradation for that bit in some sets, making cache fail earlier. Instead, with a perfectly randomized cache placement all cache lines will have 90% bias for bit 30, thus delaying the occurrence of the first failure.

Hence, we aim at finding a better *RM* implementation that balances utilization of the cache sets regardless of the particular access pattern and indexes of the addresses accessed.

4.2.2 Randomizing Set Distribution

Our proposal to make index bits have a random distribution is analogous to that used for making control bits in the Benes network be random: hashing address bits with random bits. In the particular case of the index bits, we XOR them with some random bits of the random seed. For instance, let us recall our previous example. If we XOR the index bits of 0x00 (so 0000b) with a random value, in essence we

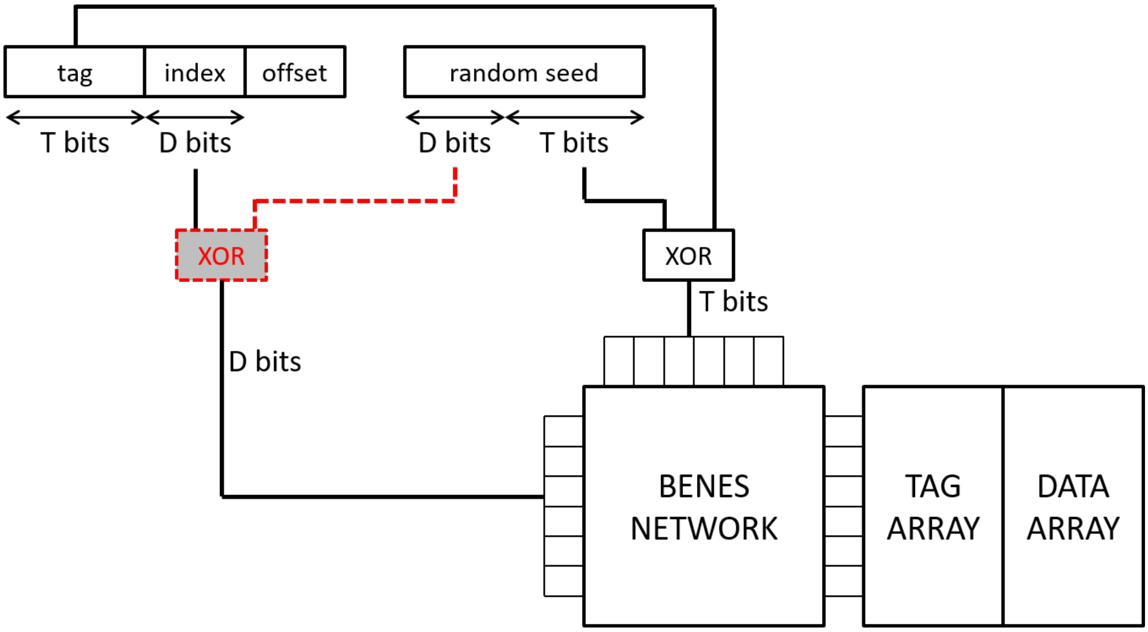


Figure 4.3: Schematic of the *enhanced* implementation of a random modulo cache. Red-dashed parts indicate the changes introduced.

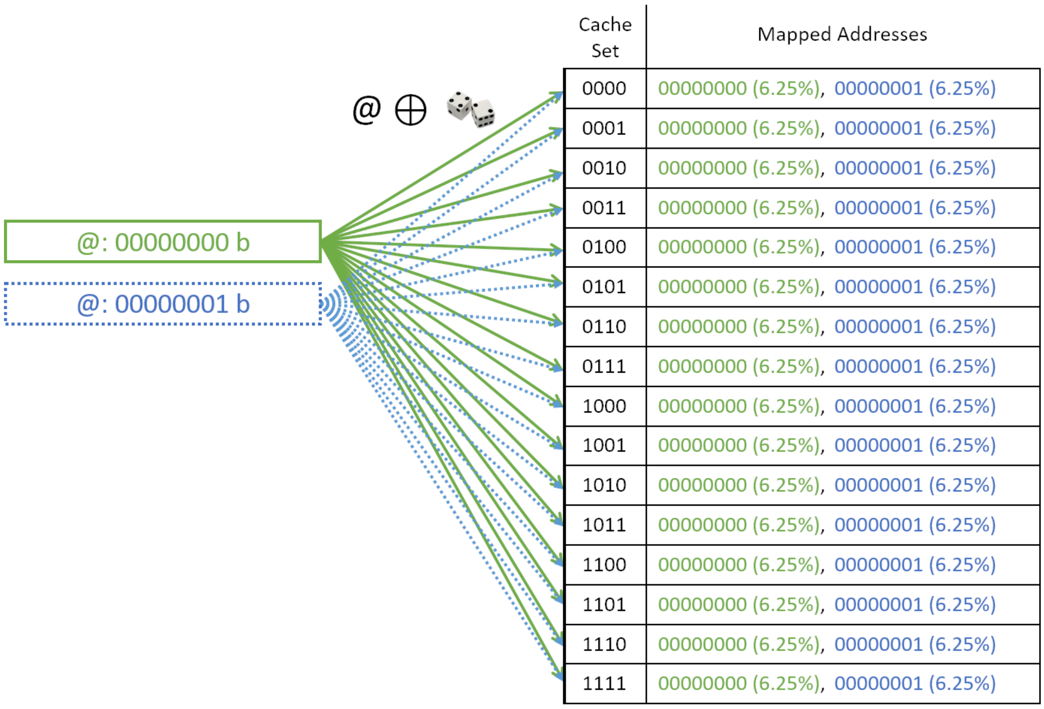
will obtain 16 different indexes – all binary values that can be encoded with 4 bits – with homogeneous probability. This also holds for any other address regardless of their particular index bits. Thus, all addresses are placed to all sets with identical probability regardless of their particular index bits. Therefore, in the long run all sets are expected to be used homogeneously regardless of the particular access patterns of the programs being run.

The drawback of this approach is that a XOR gate is introduced in the path of the index bits to the Benes network, thus potentially affecting the critical path. Still, since a single XOR gate is added, the impact is limited as proven later in the evaluation section.

Figures 4.2 and 4.3 show the baseline *RM* design and our enhanced *RM* design respectively. As shown, in our enhanced version we add a level of XOR gates to combine index bits (D bits) with D random bits taken from the random seed. In the figure we show that the random bits used for the index generation (those in the left side of the Benes network) and control bits generation (those on the top part of the Benes network) correspond to different random bits. In practice there is no constraint on using the same bits or different ones since they are used for different purposes.

For the sake of completeness, we show in Figure 4.4 the effects of implementing ERM on the example depicted in Figure 4.1. As shown, a uniform distribution is achieved with ERM regardless of the specific index bits value used as input for the indexing policy.

In summary, as shown later, this conceptually minor – but highly powerful – modification allows balancing the utilization of the cache sets, thus mitigating maximum aging and so increasing the lifetime of the cache memory. The impact in the critical



Cache Set	Mapped Addresses
0000	00000000 (6.25%), 00000001 (6.25%)
0001	00000000 (6.25%), 00000001 (6.25%)
0010	00000000 (6.25%), 00000001 (6.25%)
0011	00000000 (6.25%), 00000001 (6.25%)
0100	00000000 (6.25%), 00000001 (6.25%)
0101	00000000 (6.25%), 00000001 (6.25%)
0110	00000000 (6.25%), 00000001 (6.25%)
0111	00000000 (6.25%), 00000001 (6.25%)
1000	00000000 (6.25%), 00000001 (6.25%)
1001	00000000 (6.25%), 00000001 (6.25%)
1010	00000000 (6.25%), 00000001 (6.25%)
1011	00000000 (6.25%), 00000001 (6.25%)
1100	00000000 (6.25%), 00000001 (6.25%)
1101	00000000 (6.25%), 00000001 (6.25%)
1110	00000000 (6.25%), 00000001 (6.25%)
1111	00000000 (6.25%), 00000001 (6.25%)

Figure 4.4: Example address distribution for a 16-set cache with the enhanced random modulo design.

path is low (at most an extra XOR gate), address-to-set mapping within cache way boundaries is a permutation (thus keeping miss rates low by avoiding many potential conflicts), and MBPTA compliance is preserved since cache set location is random.

4.2.3 Hash Random Placement Set Distribution

hRP has been designed with the aim of making each address have the same probability to be mapped to each set. In the design of *hRP* Figure 2.6 (in Section 2.3.2) we can observe that the 3 leftmost rotate blocks use as input random bits, which are rotated based on some address bits. The 3 rightmost rotate blocks do the opposite: use as input address bits and rotate them based on some random bits. Overall, the output of the 3 leftmost blocks is a set of random bits, which are later XORed with the other bits, thus leading to a purely random output. Hence, regardless of the input address being accessed, its probability of being mapped to each different cache set is homogeneous in the long run. Therefore, the default *hRP* design already achieves the homogeneous set distribution of addresses for L2 caches obtained with our proposed enhanced *RM* for L1 caches. The homogeneous set distribution of *hRP* has already been proven before [71]. Note, however, that hRP, differently to RM and ERM, does not put any constraint on whether consecutive addresses can be randomly mapped to the same cache set. Therefore, even with small workloads fitting comfortably in cache, hRP can lead to high miss rates, even if with low probabilities. RM and ERM instead avoid cache misses for cache lines mapped within the same

Random Placement	Performance	Even Distribution	Page Size
hRP	-	+	+
RM	+	-	-
ERM	+	+	-

Table 4.1: Summary of the different random placement implementations and its trade-offs.

memory page. In Table 4.1 we provide a summary of the different random placement implementations and its trade-offs. Generally speaking, random modulo techniques offer better performance than *hRP* but present implementation limitations for certain cache sizes (see Section 2.3.2.2), our proposed ERM implementation improves the cache set usage distribution w.r.t. RM.

4.3 Evaluation

This section evaluates our enhanced *RM* placement and the default *hRP* and their impact on aging. First, we introduce the evaluation methodology. Then we present the results in terms of access distribution across sets, and how this can improve lifetime in terms of HCI and BTI. Finally, we show the impact of the hardware modification in the critical path.

4.3.1 Methodology

We model the first level instruction (IL1) and data (DL1) caches of a NGMP 4-core processor designed for the space domain [94]. Those caches are 16 KB 4-way 32 B/line. Thus, they have 128 sets each. We refer the reader to Chapter 3 for further details on the architecture.

We evaluate the different cache placement designs: modulo (M), default Random Modulo (RM) and our *Enhanced Random Modulo (ERM)* for L1 caches, and modulo and *hRP* for L2 caches. For our evaluation we use the EEMBC autobench suite, a well-known benchmark suite used in the real-time domain [106]. Each EEMBC benchmark is analyzed using its default input data. Considering multipath effects in the context of MBPTA has been addressed elsewhere [112] and is orthogonal to the work in this chapter.

Benchmarks have been run once in an improved version of SoCLib [99] to extract instruction and data address traces. Then, cache set distribution of each placement function has been evaluated in a cache simulator processing those address traces.

For estimating the HCI lifetime improvement, we use the expressions provided in [113] showing that transistors lifetime degradation due to HCI is inversely proportional to their switching activity. For analyzing the potential impact on BTI lifetime, we measure bit bias for all cache bits. Whether the relation of bit bias with lifetime is

Table 4.2: Distribution of accesses across sets for the different placement functions and L1 caches.

Cache	**IL1**			**DL1**		
Placement	**M**	**RM**	**ERM**	**M**	**RM**	**ERM**
`a2time`	5.5	1.5	1.0	32.0	3.5	1.0
`aifftr`	3.2	2.2	1.0	17.3	2.1	1.0
`aifirf`	5.9	1.4	1.0	27.7	10.9	1.0
`aiifft`	2.4	1.5	1.0	17.2	2.1	1.0
`basefp`	7.4	1.8	1.0	36.6	4.1	1.0
`bitmnp`	2.1	1.4	1.0	26.8	2.6	1.0
`cacheb`	12.7	2.6	1.0	24.8	2.3	1.0
`canrdr`	11.1	2.0	1.0	36.4	5.7	1.0
`idctrn`	3.0	2.9	1.0	26.7	2.8	1.0
`iirflt`	7.7	1.3	1.0	20.6	4.2	1.0
`pntrch`	3.9	1.8	1.0	30.9	3.4	1.0
`puwmod`	19.1	2.7	1.0	63.3	3.2	1.0
`rspeed`	8.5	2.1	1.0	41.7	3.6	1.0
`tblook`	4.2	2.0	1.0	22.1	4.2	1.0
`ttsprk`	18.6	2.1	1.0	53.8	7.1	1.0
HARMEAN	4.9	1.8	1.0	27.9	3.4	1.0

linear (self-healing effect is negligible) or exponential (self-healing is significant) does not change the conclusions reached analyzing bit bias only.

To quantify delay overheads of the *ERM* implementation we have described both circuit implementations, the original *RM* and the enhanced one, with VHDL and synthesized them using Synopsys DC [102] with a TSMC 45nm technology library [103]. Additionally, both implementations have been integrated in a 4-core Leon3-based processor resembling the NGMP and synthesized in a Stratix IV Altera device at 100 MHz.

4.3.2 Set distribution

First we evaluate the access distribution across sets for each cache (IL1, DL1 and L2) and placement function: *Modulo*, *Random Modulo* and our *Enhanced Random Modulo* for L1 caches, and *Modulo (M)* and *hRP* for L2. For each one we show the ratio between the maximum number of accesses per set and the average number of accesses per set (MAX/AVG). In the ideal case where accesses are perfectly balanced, MAX/AVG should be exactly 1. In general we can expect MAX/AVG to be higher than 1.

Table 4.2 summarizes the results for all benchmarks for L1 caches. We use $100,000$ runs with different random seeds for *RM* and *ERM*. Since *M* always delivers the same distribution, one run suffices. As shown, *M* produces high imbalance across sets, particularly for the DL1 cache. The particular addresses accessed determine

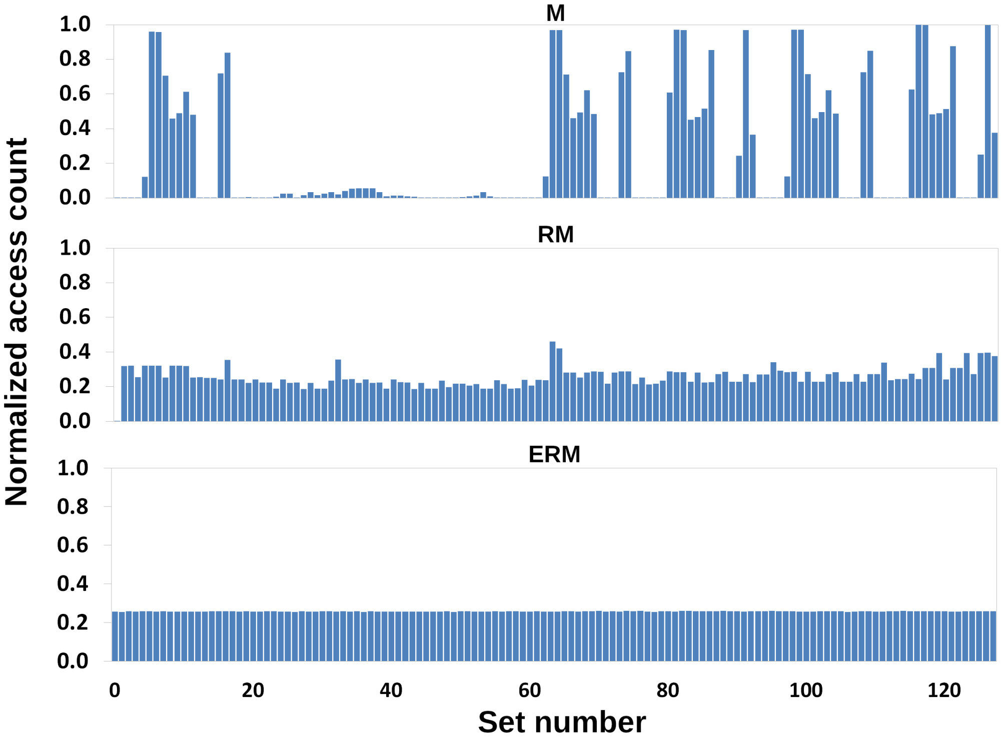


Figure 4.5: IL1 per-set access distribution for `pntrch`. Values are normalized with respect to the number of accesses of the maximum accessed set in the modulo configuration.

the sets accessed, and so the set distribution. Thus, M distribution is completely program-dependent. This is in part mitigated for the IL1 since loops contain some significant sequential code accessed many times, thus leading to quite homogeneous distribution (at least for the sets accessed in the loop). Conversely, access patterns for DL1 can be highly irregular in many cases, thus leading to high imbalance in the set access distribution. The harmonic mean for the set distribution of M is 4.9 for the IL1 and 27.9 for the DL1, thus far from the ideal value 1.0.

RM balances accesses much better due to the randomness introduced in the generation of the set index. This is particularly noticeable for the DL1. Still, since some dependence exists between the actual addresses accessed and the sets where they map, set distribution improves only to some extent. The harmonic mean for the set distribution of RM is 1.8 for the IL1 and 3.4 for the DL1. While these results are far better than for M, they are still far from the ideal value 1.0, especially for the DL1.

Finally, our ERM removes the dependence of the set index on the particular address accessed, thus delivering much better set access distributions. This effect is particularly relevant for the DL1, where the imbalance for both M and RM is high. The harmonic mean for the set distribution of ERM is 1.0 (Max:1.028) for the IL1 and 1.0 (Max:1.044) for the DL1, very close to the ideal value 1.0. This translates into marginally better average performance (below 0.1% performance improvement).

For completeness, we show the per-set distribution for the different L1 placement

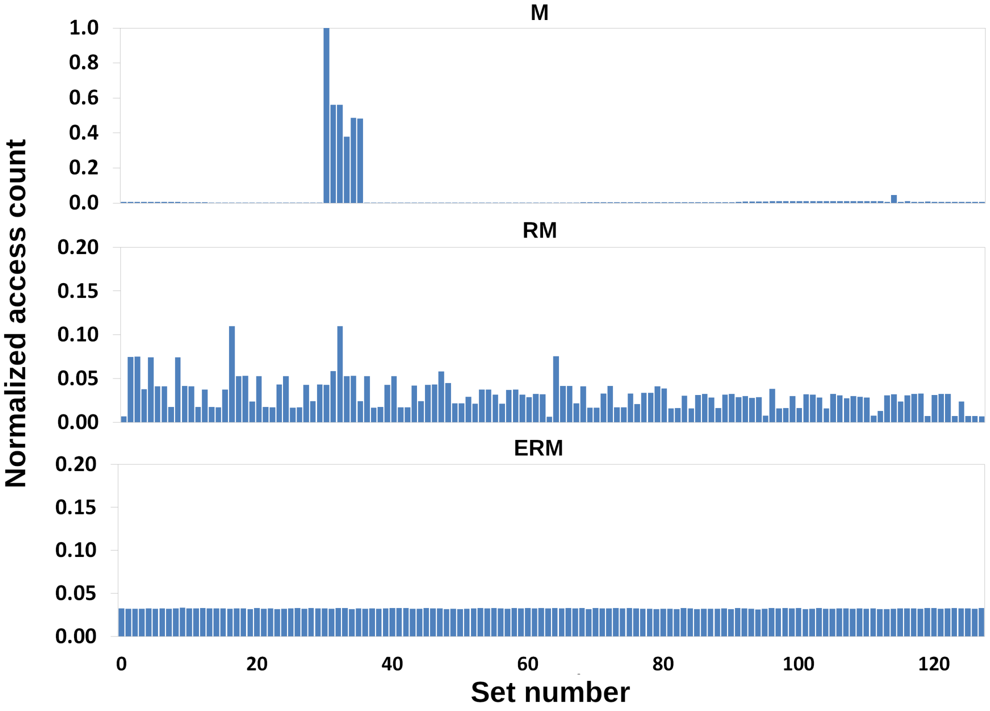


Figure 4.6: DL1 per-set access distribution for `pntrch`. Values are normalized with respect to the number of accesses of the maximum accessed set in the modulo configuration. Note the different scales in each configuration.

policies for 2 specific examples: the IL1 and the DL1 for `pntrch`. The former (see Figure 4.5) is a relatively good case for M, whereas the latter (see Figure 4.6) is a relatively bad case for M. Figures show in the x-axis the different cache sets and in the y-axis the utilization normalized with respect to the highest utilization in the M case.

The example in Figure 4.5 shows that M uses some sets quite often, whereas others are barely used. Still, the number of sets used often is relatively large. RM achieves a much better distribution across sets and only some sets have higher utilization than the average. Those sets correspond to those with most index bits being zero (or one), so that randomization has limited effect. Finally, our ERM achieves almost homogeneous cache set utilization.

The example in Figure 4.6, instead, shows that M uses very few sets of the DL1 cache for `pntrch`. This leads to an extremely unbalanced distribution. In the case of RM (note that the y-axis only reaches 0.2) the distribution is far better as the most used set is used around 8 times less than for M. Still, unbalance is high. Finally, ERM achieves almost homogeneous set utilization.

When analyzing set distribution for the L2 cache (see Table 4.3), we observe that the set distribution for M is highly biased, ranging between 99.7 and 866.2 across benchmarks, being much worse than for L1 caches. This occurs due to two combined effects: (1) the number of sets in L2 is much larger, thus increasing the chances that highly used sets are still highly used whereas many more sets remain mostly unused;

Table 4.3: Distribution of accesses across sets for the different placement functions in the L2 cache.

Cache	**L2**	
Placement	**M**	**hRP**
`a2time`	679.9	1.17
`aifftr`	160.3	1.08
`aifirf`	396.7	1.17
`aiifft`	159.9	1.08
`basefp`	712.6	1.16
`bitmnp`	99.7	1.09
`cacheb`	353.1	1.15
`canrdr`	772.8	1.23
`idctrn`	631.3	1.18
`iirflt`	583.9	1.16
`pntrch`	294.8	1.12
`puwmod`	866.2	1.19
`rspeed`	638.1	1.17
`tblook`	651.6	1.18
`ttsprk`	829.8	1.20
HARMEAN	339.4	1.15

and (2) L1 caches filter many accesses to L2, which typically increases the imbalance across sets. hRP instead achieves a highly homogeneous distribution across sets with an harmonic mean of 1.15 across benchmarks and a maximum of 1.23 for `canrdr` benchmark. Moreover, hRP imbalance progressively approaches 1.0 as we increase the number of runs, which is the expected value after an infinite number of runs due to the purely random address-to-set mapping nature of hRP.

4.3.3 Lifetime

Next we elaborate on the impact of the improved set distribution on HCI and BTI lifetime.

HCI. We use the HCI model reported in [113], as explained before, where it is shown that HCI is directly proportional to the switching activity. Thus, we use actual switching activity of the cache accesses to estimate the relative HCI lifetime improvements. We assume that a failure occurs when the first permanent fault due to HCI occurs. Thus, the bit with highest switching activity determines cache lifetime. While other models could be used, our work is centered around safety-related real-time systems where timing verification occurs before operation and assuming that the processor is fault-free. Thus, unless otherwise considered during the analysis phase, one faulty cache line may impact the WCET and thus, invalidates those timing guarantees on which the certification process has been conducted.

Since the actual lifetime value depends not only on HCI, but also on other sources

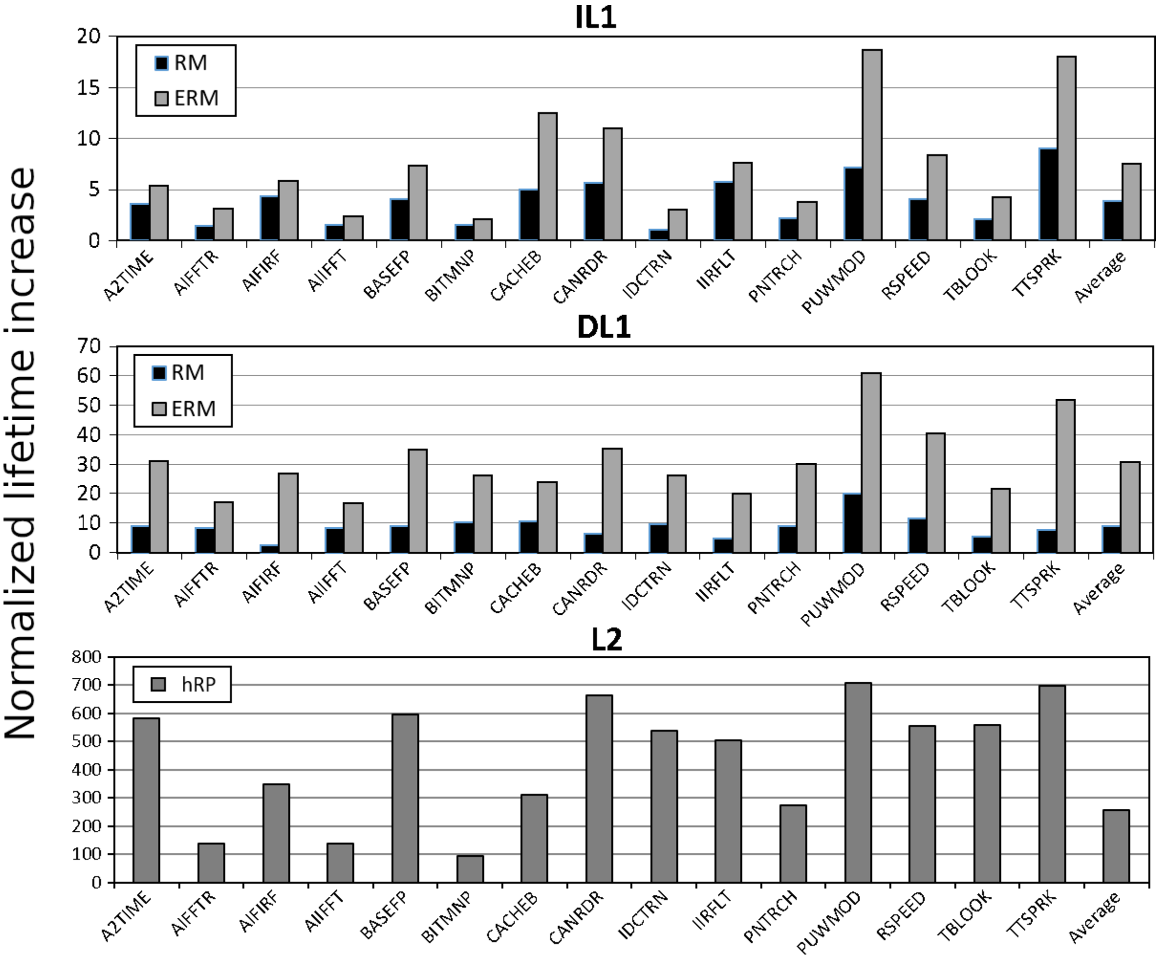


Figure 4.7: Increase in cache lifetime due to HCI impact. *RM* and *ERM* configurations normalized with respect to *M* for both IL1 and DL1, and *hRP* configuration normalized with respect to *M* for L2. Note that scales are different in each plot.

of failure and hardware components, we report how much the lifetime of the IL1 and DL1 is extended due to HCI for *RM* and *ERM* normalizing the results with respect to *M* placement. Results are shown in Figure 4.7. We observe that lifetime grows by 3.9x and 8.8x on average for IL1 and DL1 respectively for *RM*. Results for *ERM* are far better, extending HCI lifetime by 7.6x and 30.9x on average for IL1 and DL1 respectively. If we compare *ERM* with respect to *RM*, lifetime grows by a factor of 1.9x and 3.5x for IL1 and DL1 respectively.

Regarding the L2 cache, we observe that *hRP* extends L2 lifetime by a factor of 257x with respect to *M* placement, which is a huge gain. Still, one must consider that HCI is proportional to switching activity and L2 cache lines are typically read/written only a fraction of the times L1 caches are. Therefore, if L1 and L2 caches use the same type of transistors, L1 caches experience much higher switching activity and thus, L1 caches are the reliability bottleneck for HCI. If, instead, L2 caches are implemented with smaller transistors for achieving further integration, then L2 caches may be the reliability bottleneck for HCI, particularly if DL1 is write-through so that all store operations are forwarded to L2 cache. Note that write-through DL1 caches are common in the safety-related domain due to the complexity to implement complex error correcting codes in DL1 caches without impacting cycle time or cache latency [94]. Instead, these latencies are better tolerated in L2 caches and L1 caches can implement parity instead since error correction can be performed using L2 cache contents.

BTI. As explained before, BTI aging does not relate to the switching activity but to the content bias of the cache cells. In order to understand the benefits of *ERM* in L1 caches and *hRP* in the L2 cache, we have performed the following experiment for the IL1: we collect the bit bias for each cache line bit considering the actual switching activity for the EEMBC benchmarks and disregarding the effect of replacement policies, therefore using a 1-way IL1 since random replacement already exhibits a fair random distribution and we focus just on the placement. Note that random replacement, as well as hRP and ERM, are fully random. Hence, what cache line is replaced in a cache with N sets and W ways is randomly chosen and homogeneous across $N \cdot W$ lines. This is also the case for a cache with $N \cdot W$ sets and 1 way. In the case of RM (or M), the actual cache set selected is not fully random and homogeneous. As we increase the number of ways and decrease the number of sets (e.g., 2x ways, 0.5x sets), using random replacement, the bias introduced by the placement policy would get increasingly mitigated. Thus, we evaluate direct mapped caches to exacerbate differences. We have performed this experiment for all benchmarks and obtained similar results across all of them. While this experiment provides only a first-order approximation to the impact of *ERM* on BTI, our conclusions would hold with any BTI lifetime model in the light of the results presented next. Figure 4.8 shows in the top part the bit bias obtained for each bit of each set (i.e., a line in a 1 way cache) in the IL1 for M placement for `pntrch` EEMBC benchmark, where lighter bits indicate they are highly biased (i.e., storing a particular value for most of the time) towards '1' whereas darker bits are highly biased towards '0'. As shown, many bits are very highly biased towards specific values. In fact, 235 out of the 256 bits per cache line are 100% biased towards '0' or towards '1' in at least one of 128 cache sets. Thus, independently of how much time each cache line is stored in cache and how the replacement policy distributes cache lines across ways, there will be some bits 100% biased towards '0' or '1', thus experiencing maximum degradation. Note that the distribution is heterogeneous across sets, thus reflecting the fact that the particular sets used with M placement directly depend on the memory addresses where objects (code in this case) are stored.

We have evaluated what *ERM* could achieve in terms of bit bias considering actual switching activity, also disregarding the effect of replacement policies as for M placement. The bit map, again for `pntrch`, is shown in the bottom part of Figure 4.8. As shown, *ERM* homogenizes the bias across sets (visually across rows), but cannot do anything to further reduce the bias. This would bring some gains in terms of BTI lifetime since the maximum bias across all bits in cache decreases from 100% to 80%, which would extend lifetime. How much lifetime would be extended would depend on the actual technology and the degree of self-healing it achieves. Lifetime gains would be proportional to the maximum bit bias reduction if self-healing is neglected.

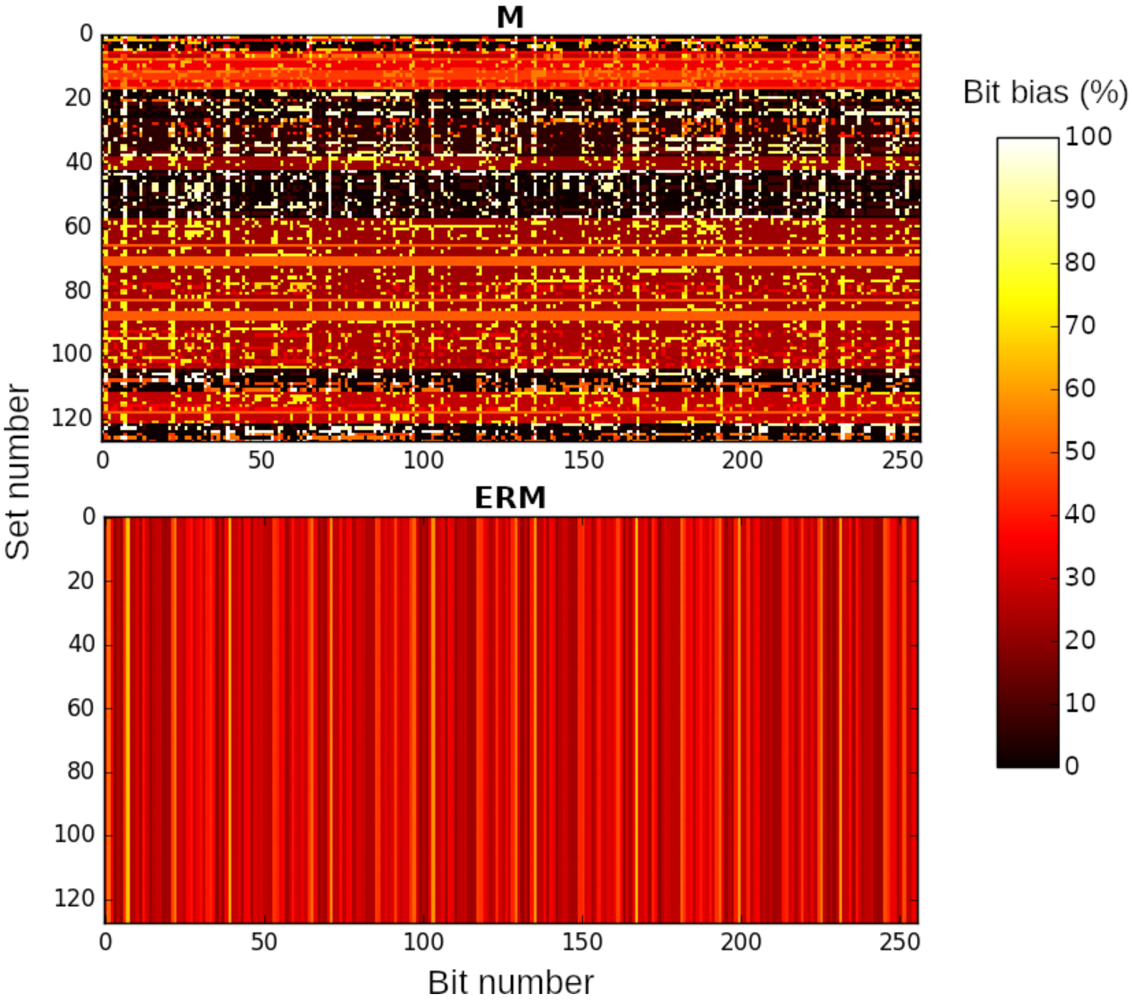


Figure 4.8: Bit bias for the IL1 cache with *M* (top) and *ERM* (bottom) placement policies for the `pntrch` EEMBC benchmark. Lighter color indicates higher bias towards '1' while darker color indicates higher bias towards '0'.

Considering self-healing would affect those results, which could be potentially optimistic. In any case, the bit bias achieved is still far away from the ideal 50%. Similar conclusions are reached for *ERM* in DL1 and *hRP* in L2 cache across all benchmarks since bias in those caches reaches 100% for *M*, but remains far away from the 50% for randomized designs. Hence, caches implementing some form of random placement are expected to achieve higher BTI lifetime than those implementing modulo placement based on our bit bias measurements, but are far from the lifetime gains that could be achieved by fully balancing bit bias with techniques such as those in [114, 115, 116] based on inverting cache contents. This effect is expected since random placement homogenizes bit bias across cache sets, but cannot do anything if specific bit positions are highly biased towards one value.

Therefore, we can conclude that, while randomized caches could provide some benefits in terms of BTI lifetime, other orthogonal techniques based on content inverting are likely needed to mitigate BTI aging if it is a reliability bottleneck.

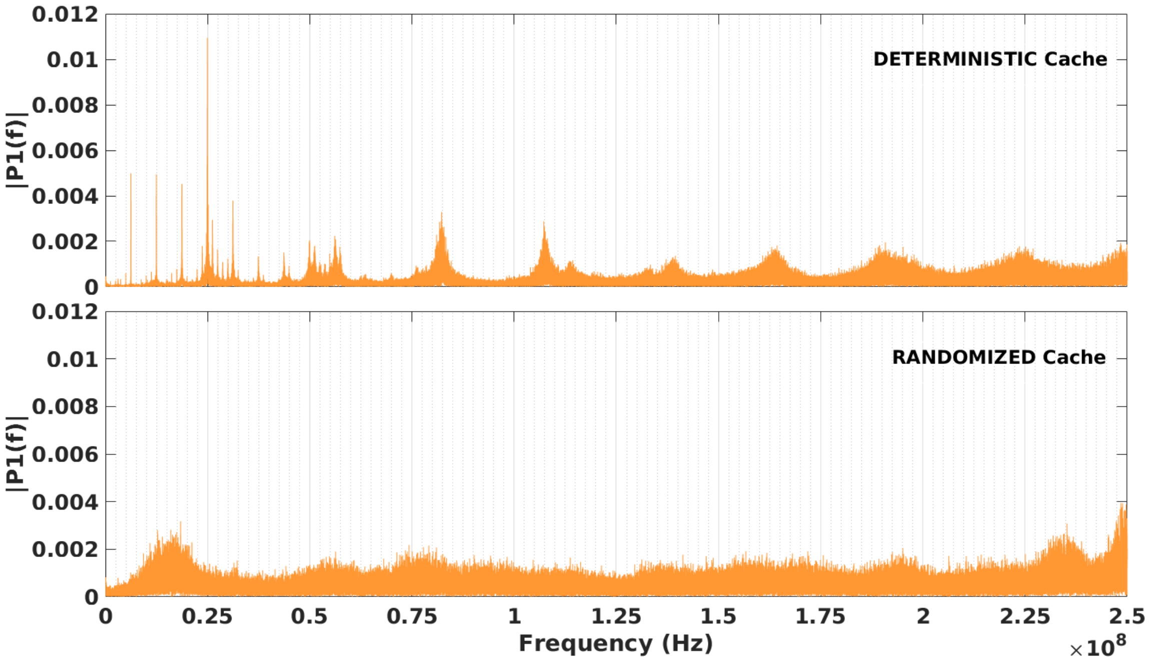


Figure 4.9: Frequency spectrum of power dissipation for a regular processor (top) and a time-randomized processor (bottom).

4.3.4 Delay impact

We have synthesized our *ERM* design using a TSMC 45nm technology library to measure its impact on cycle time. After synthesis we have seen *ERM* has zero impact with respect to the maximum operating frequency that the regular *RM* implementation can achieve. *RM* critical path depends on the complexity introduced for XORing and combining TAG bits and random seed bits to configure the Benes network. With the implementation described in [37] the critical path for both *ERM* and *RM* is 1.01 ns whereas the path affected by the inclusion of the XOR gate has 0.22 ns delay in the case of *ERM* (0.09 ns for *RM* plus 0.13 ns due to the level of XOR gates added). While faster *RM* implementations could be devised, they require at least two levels of XOR gates to combine TAG and random bits, thus being 0.26 ns the lowest potential delay (still higher than 0.22 ns).

4.4 Resonant Voltage Noise Resilience

Interestingly, randomization can be in fact also beneficial against another reliability related problem, the RVN phenomenon. RVN are pathological energy consumption patterns induced by the power demand behavior of instructions and processor events. The alignment of this events at Power Delivery Network (PDN) resonant frequencies can create power droops that might hinder processor's correct operation.

In conventional processor architectures this pathological scenarios can lead to the systematic occurrence of recurrent high power demanding events. In processors including randomization the occurrence likelihood of extreme situations is not only reduced but also it can be measured quantitatively using statistical tools, which allows to properly dimension the system [80]. Figure 4.9 shows the frequency spectrum resulting from the application of the Fast Fourier Transform (FFT) to the power profile of a pathological benchmark executed in a conventional processor (top) and in a time-randomized one (bottom). As shown in the plot, in a time-randomized processor the synchronization of power demanding events is much less significant leading to an almost flat spectral behavior.

4.5 Related Work

Several approaches have been proposed in the literature to address the impact of HCI and BTI in processors and cache structures [116, 117, 115]. While the majority of works focus on BTI effects, some recent works have also pointed out the importance of considering the effects of HCI degradation [118, 113, 114]. In fact, authors in [114] have already proposed improving the uniformity of accesses to the cache to mitigate HCI and Negative-Bias Transistor Instability (NBTI) degradation. Unlike in [114], where authors rely on introducing dedicated hardware resources to achieve uniform access distribution, we rely on the good properties of random modulo [37] and hash-based random placement [31, 110, 71] to achieve the same goal at roughly no cost.

While different works offer divergent views on the benefits of the self-healing effect for BTI, this has no impact on the conclusions of our work, but on the actual lifetime gains that could be obtained. In particular, some authors show that, whenever transistors are not degrading, their degradation rolls back progressively to some extent [79]. Other authors, however, notice that such recovery rolls back very quickly when the transistors are stressed again, thus questioning to what extent this recovery extends the lifetime of transistors [119]. Nevertheless, the best and worst case for SRAM cells do not change regardless of the benefits of the self-healing effect. Hence, in our particular study we stick to a first-order approximation for BTI effects relating BTI degradation to stress duration, thus disregarding the self-healing effect. As shown later in our evaluation, our conclusions would not be affected by this effect if taken into account.

Considering WCET estimation together with reliability issues has been done at several fronts. Some authors propose preserving WCET estimation methods by devising hardware cache designs able to tolerate permanent faults with no effect (or easy to account effect) on WCET estimates [22, 120]. Other authors propose accounting for the timing impact of faults in a probabilistic manner in combination with static deterministic timing analysis methods by studying the impact and probabilities of different fault distributions [20, 121, 21]. However, since those approaches do not use randomized cache designs, the impact estimation of (random) faults has to rely on the most conservative assumptions. Additionally, they cannot be applied in the context of measurement-based timing analysis.

Recently, some authors have done some preliminary work in the context of WCET analysis of faulty hardware together with MBPTA [111, 23]. Results are promising and prove that the random nature of the timing of hardware providing the properties required by MBPTA matches very well with the random nature of faults, thus leading to efficient solutions. While that work shows to be efficient to account for the impact of transient faults and a limited number of permanent faults, it does not provide means to mitigate aging effects.

In this chapter, instead, we assess the reliability of the different random placement designs, random modulo for L1 caches and hash-based random placement for L2 caches, in terms of aging (HCI and BTI). Then, we propose an enhanced random modulo implementation such that reliability of L1 caches is improved while preserving the good properties of random modulo.

4.6 Summary

Fault tolerance and WCET estimation, needed both for safety-related real-time systems verification, have often been addressed as separate concerns. In this context, approaches based on MBPTA have been shown to match very well the needs of both concerns by relying on the same principle: randomness. Therefore, efficient solutions can be built to consider both concerns simultaneously.

In this chapter we assessed the reliability in terms of HCI and BTI of random modulo cache designs for L1 caches and hash-based random placement for L2 caches, proven convenient for MBPTA. Our results show that random placement designs effectively mitigate HCI aging and provide some limited benefits in terms of BTI. Still, random modulo is far from being optimal in terms of HCI. Therefore, we propose an alternative random modulo implementation that further mitigates HCI aging while preserving the main features of random modulo: low impact in critical path, low miss rates and adherence to the requirements of MBPTA. Additionally we also demonstrated how TRP are naturally more resilient to pathological effects like RVN.

Part III

Security in CRTES

Chapter 5

Attaining Side-Channel Attack Resiliency and Time-Predictability

> *"It is possible to provide security against other ills, but as far as death is concerned, we men live in a city without walls."*
>
> — Epicurus

5.1 Introduction

Security has become a primary concern for computing systems in the last decades with a plethora of types of attack already proven successful. For instance buffer overflow, double freeing or Spectre [122] attacks have demonstrated security flaws in the design of software and hardware.

In this chapter we analyze how time-predictability (as an example of safety concern) and side-channel attacks (as an example of security issue) in cache memories can be jointly tackled and later extend on how randomization can also be used against other security intrusions.

Increasingly autonomous and connected Automotive Systems (ATS) require on-board computing systems with resilient operation under timing faults and attacks. The increased connectivity of these systems opens the door to security threats (e.g., side-channel attacks, an effective security intrusion for obtaining secret keys). In particular, Cache-Timing Side-Channel Attacks (SCA) (see Section 2.4.2.3) allow attackers to fully or partially recover keys, which can later be used to take control over the ATS. On the other hand, ATS increasingly deal with safety (e.g., autonomous driving), which requires a reliable response time of all critical software services.

Until recently, ATS comprised relatively simple Electronic Control Unit (ECU) deploying 8 and 16 bit microcontrollers. The execution time of software on such simple devices was mostly jitterless (or suffering very small execution time variability) simplifying the task of deriving Worst-Case Execution Time (WCET) estimates and mitigating the risk of SCA. However, the advent of more complex value-added software functionalities, managing increasing amounts of diverse data, has raised the performance needs for the automotive sector.

The execution time of tasks on complex hardware strongly depends on their input data and processor's state, thus exposing the system to SCA. This is a major concern for ATS to protect sensible information and prevent safety issues as ATS control critical aspects with humans in the loop like autonomous driving.

Interestingly, randomization has been independently proposed as a solution for WCET estimation and preventing SCA. For WCET estimation, the most extended industrial practice builds on collecting execution time measurements of the software running on the target platform. Obtaining evidence about whether those measurements are representative of the WCET during operation is challenging on complex hardware [13].

These difficulties have been addressed by using probabilistic techniques to WCET analysis, so called Measurement-Based Probabilistic Timing Analysis (MBPTA) techniques [34]. MBPTA benefits from injecting randomization in cache timing to simplify modeling and provide evidence for certification as needed by safety standards. For SCA, randomization solutions break the dependence between input data (and/or cache state) and execution time so that for the same input data and processor state a sufficiently random execution time is experienced. However, it remains to be proven whether a single solution can tackle both, WCET and SCA issues.

While injecting randomization in cache timing-behavior addresses each of those concerns separately, we show that randomization solutions for time-predictability do not protect against side-channel attacks and vice-versa. We then propose a randomization solution to achieve both safety and security goals.

5.2 MBPTA and SCA Properties

To understand the inherent properties that each of the domains (safety and security) demand we first make a separate analysis of the desired characteristics for MBPTA and security.

5.2.1 Random Caches for MBPTA

Often, in the automotive domain, system integrators subcontract the development of certain software to different software providers. Across software integrations of the software units contributed by each provider, the objects of a function (i.e., globals, stack and code) can change their addresses resulting in different cache layouts, with arbitrarily different hits/misses and execution time [123]. In general, it is unfeasible for users creating test scenarios during the Analysis Phase (AP) accounting for the worst memory placement (and cache layouts) that can occur during system's Operation Phase (OP) [13].

Cache randomization [37, 124] ensures that a new random cache layout is exercised on every program run so that the impact of caches on execution time becomes independent of the actual memory layout. This relieves users from controlling memory/cache layout of objects since (random) cache layouts experienced during AP and OP are probabilistically identical. MBPTA builds upon collecting a sample of execution times of the task under analysis on the target platform and verifying that those samples meet certain statistical properties in order to properly apply statistical tools to enable timing analysis, see Figure 2.4 (left). In particular, MBPTA applies Extreme Value Theory (EVT) [34] that requires independence and identical distribution of the execution times [34] (see Section 2.2.1).

Cache randomization involves three main elements: a Pseudo-Random Number Generator (PRNG), random placement and random replacement, the latter of which is optional. Several works show the existence of low-overhead PRNG that provide enough quality in the sequences produced to avoid correlations [125]. Regarding random placement [31, 37, 124], it requires to adhere to certain properties for MBPTA compliance (referred to as *mbpta-px*).

mbpta-p1 *Time composability* across incremental software integration ensures that early phase WCET estimates (ideally at the unit testing level) hold upon integration. This decreases the risk of costly detection of timing violations during late design phases. Time composability, relates cache layouts (i.e., how addresses are mapped to cache sets) during AP and OP. Time composability builds on one of the following properties on random cache placement.

mbpta-p2 *Full Randomness.* Let us assume two different addresses A and B (i.e., they differ at least in one bit, excluding offset bits within the cache line) and a cache with S sets.

1. A (and B) is randomly placed to different sets for different seeds: That is, there exist seeds $seed_i$, $seed_j$ and $seed_k$ so that $S_A^{seed_i} \neq S_A^{seed_j}$, and $S_A^{seed_i} = S_A^{seed_k}$.
2. The set where A and B are mapped to is not systematically identical. That is, for some seeds, $seed_i$, $S_A^{seed_i} \neq S_B^{seed_i}$, whereas for others, $seed_j$, $S_A^{seed_j} = S_B^{seed_j}$.
3. It is required to keep the same cache-line alignment during AP and OP so that if A and B belong to different cache lines at OP, they also do in experiments carried out at AP.

Note that, for any seed, it is not needed that A and B have the same probability of being mapped to the same set.

mbpta-p3 *Partial APOP-fixed Randomness.* In this case, randomization is carried out at memory page boundary.

1. If A and B are in the same page boundary, the probability to map them to the same set is null for any seed.
2. If A and B belong to different pages, the same principles than for full-randomization apply.
3. It is required to keep the same memory-page alignment during AP and OP so that if A and B belong to different pages at OP, they also do in experiments carried out at AP.

5.2.2 Random Caches for Security

SCA extract secret key data by exploiting the information leakage resulting from the physical implementation of the system. An important bulk of SCA exploits the information leakage of the hardware through input data dependent execution time. The most common timing-related SCA exploit cache behavior to obtain sensitive information. There are several main types of SCA able to exploit different cache properties, but in this thesis we focus on the contention-based SCA.

Contention attacks are based on conflicts between cache lines, and include Prime-Probe and Evict-Time attacks. The basic concept is to use caches to perform or expect evictions of cache lines, and from there infer which data is being utilized. The attack vector is enabled because cache lines are stored in sets in a deterministic fashion.

Given that accesses to the lookup tables depend on the input data, an attacker is able to extract cryptographic keys by measuring the time it takes to the victim or itself to load certain data. Most of this kind of attacks assume that the attacker shares the use of the processor with the victim [81]. However, it is not necessary to have a contender in the same processor in order to perform contention attacks. Bernstein [41] for instance, proved that interference inside the victim's own accesses might be enough to reveal full cryptographic keys.

Let Ω be the universe of input states, Γ the universe of execution times that a task can exhibit, t_ω^γ the execution time $\gamma \in \Gamma$ of a cryptographic task given the input $\omega \in \Omega$, and $P(t_\omega^\gamma)$ the probability of observing such execution time.

For protection against SCA, **sca-p1** cache designs must ensure that no correlation exists between the input data and the observed execution time. In this way, any single input state can exhibit several execution times with equal probability, thus preventing any attacker from identifying the input state from the execution time:

$$\forall i \in \Omega, \forall g \in \Gamma \mid P(t_i^g) = \frac{1}{\mid \Gamma \mid} \tag{5.1}$$

5.3 Assessing the Time-Predictability of Secure Cache Designs

The **RPCache** [126] decouples cache interference of the attacker from the victim by randomizing interference whenever a memory access from a process different from the victim's one contends for the same cache line. On the event of a contention event that might leak information, a random set is selected for replacement. So far the MBPTA compliance of this design has not been assessed. In particular we identify two features of this approach that fail to meet MBPTA requirements: First, the timing behavior of the task under analysis depends on the actual addresses accessed. Therefore, WCET estimates do not hold across integration with other software units, which may change the actual addresses of the task and hence, change its timing behavior arbitrarily. This prevents achieving time composability (requirement mbpta-p1). And second, contender tasks produce cache evictions in random sets upon contention with the task under analysis. Hence, since whether contention exists is fully dependent on task's contenders behavior, so is the case for task's cache evictions. Thus, the WCET estimate obtained for the task strongly depends on the contenders behavior, typically unknown in early design phases (when WCET estimates need to be successfully assessed against timing budgets). This is against time composability needed in ATS (mbpta-p1), as explained before.

The **Newcache** [127] builds on the same concept as RPCache and introduces several improvements to reconcile high-performance and security, offering low miss rates and faster accesses. However, the main concept and limitations for MBPTA-compliance behind the placement policy remains the same as for RPCache.

Aciicmez [128] proposes a placement policy to secure instruction caches that randomizes the cache set where addresses are placed by XORing the index bits with a random number. Let's assume two addresses A and B and further assume that they have identical index bits. Hence they are placed in the same set with modulo. By XORing their (identical) index bits with a random number, the set obtained is random, but identical for both addresses, hence breaking mbpta-p2 principles. Furthermore, if A and B have different index bits they are placed in different sets with modulo. By XORing their (different) index bits with a random number, the set obtained is also different, so they are placed in different sets. This breaks mbpta-p2 since it is not the case that for different seeds $seed_i$ and $seed_j$: $S_A^{seed_i} \neq S_B^{seed_i}$ and $S_A^{seed_j} = S_B^{seed_j}$.

While [128] performs a random permutation of cache sets based on the random number used, its timing behavior is strictly dependent on the addresses and analogous to that of modulo placement, breaking mbpta-p1. Also, performance (and so WCET) are strictly dependent on the actual addresses accessed, so WCET estimates become invalid upon integration since different memory locations of objects will lead to arbitrarily different cache conflicts.

5.4 Assessing the SCA-robustness of Time-Predictable Cache Designs

For MBPTA compliance, caches implement random placement and (optionally) random replacement. The latter, on the event of a miss in a given cache set, randomly chooses one line in the set to be evicted and replaced. As explained before, random replacement builds on PRNGs so that choices are sufficiently random [125]. Random placement, determines the cache set where an address is mapped by operating the address (tag and index bits) together with a random number called random seed or just *seed* [31, 37, 124]. The remaining address bits (offset bits) are only used to select the particular word accessed within the cache line. Given an address and a *seed*, random placement delivers a fixed cache set. However, differently to the proposal by Aciicmez [128], addresses are placed randomly and *independently* in cache. For instance, addresses A and B are placed on the same set for some seeds only. Hence,

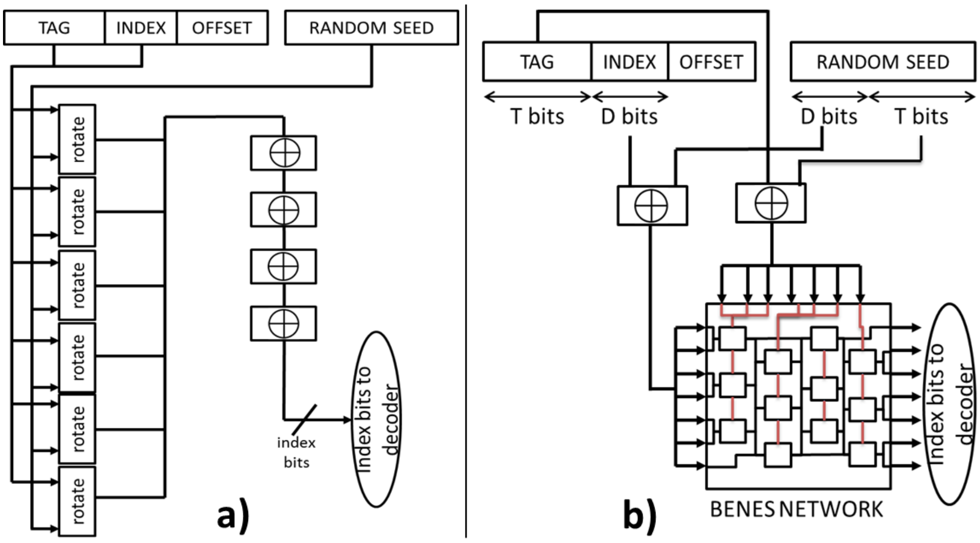


Figure 5.1: (a) hRP and (b) ERM cache architectures.

cache conflicts across different seeds are random, making actual addresses irrelevant when determining the cache sets they are mapped to. This relieves the end user from controlling memory mapping. Two different designs implement random placement: hash-based parametric random placement (hRP) [31] and random modulo (RM) or enhanced random modulo (ERM)[1] [37, 124], see Figure 5.1.

hRP [31] operates on tag+index address bits with a *seed* by means of rotator blocks and XOR gates so that conflicts in different sets are made random, see Figure 5.1 (a). hRP poses no constraint on whether cache lines need to belong to the same page. Its performance is slightly lower than that of ERM and modulo placement, but it is compatible with second level (L2) and third level (L3) caches whose way size may be much larger than the page size. hRP achieves *Full Randomness* (mbpta-p2).

ERM[1] [37, 124] takes as input the XORed bits of the *seed* with the index and tag bits of the address, see Figure 5.1 (b). The XORed index bits are the input to a Benes Network [129] and the XORed tag bits are used to drive the network. The output of this Benes Network is a randomized permutation of the index bits that point to a particular cache set. ERM is compatible with caches whose page size is equal or a multiplier of the cache way size. Often first level (L1) caches use a way size equal to the page size. With ERM each address is placed in a random set with uniform probability, but addresses in the same page are placed in different cache sets. ERM is compatible with MBPTA when contents in each memory page are preserved across integrations (easily achieved by current RTOS), while allowing pages move freely in the memory space. Overall, ERM achieves *Partial APOP-fixed Randomness* (mbpta-p3).

Compliance with SCA protection properties. MBPTA compliance for caches

[1] Without loss of generality, in this chapter we will be using ERM, the enhanced version of RM (see Chapter 4 for more details on its implementation).

relies on random placement to exhibit randomized execution times. To achieve SCA robustness, random placement must also decouple cache interference of the attacker from the victim. That is, memory addresses from victim and attacker's processes must not contend systematically in the same cache set. Instead, each memory address from each process must be randomly and *independently* placed in a set, thus randomizing interference. In the following section we detail how to achieve time-predictability and security against SCA in the same design.

5.5 Time-Secure Caches

In order to attain both, MBPTA-compliance and SCA robustness, either MBPTA-randomization solutions are made SCA-aware or SCA-randomization solutions are made MBPTA-aware. Without loss of generality, we opt for the former. SCA-aware caches cause variations in timing behavior for which achieving MBPTA-compliance require specific ad-hoc solutions. Studying those solutions is part of our future work.

hRP and ERM preserve the same *seed* during the execution of a task, so that cache contents can be retrieved and kept consistent. When cache contents are private to each task (there is no shared data), then flushing is not needed across different tasks despite using different seeds since coherency is not affected (across different tasks). Whenever a given task instance (i.e., job) executes, then either cache contents need to be flushed or the *seed* used in the previous job of the task has to be used again to preserve coherency.

MBPTA-compliance adds light constraints on seed management. Depending on the scope of the application of the WCET estimate, which for instance defines whether exceedance thresholds apply to the task as a whole or to each job independently, the granularity at which the seed has to be changed varies. On one extreme of the spectrum the seed is (randomly) set once before the execution of the first job of a task. On the other extreme of the spectrum the seed is changed right before every job release.

Interestingly MBPTA-compliance sets no constraint on the seeds used for different software units (tasks), which threatens security since two different tasks could have the same *seed* and therefore their behavior can be reproduced.

In the context of SCA, for contention attacks if the attacker task is allowed to use the same *seed* as the task under analysis, then it will have the same (random) placement as the victim. Hence, it will have the ability to learn about the victim as we show later in the evaluation section, using contention-based attacks. Instead, if each task is forced to have a different *seed*, conflicts between attacker's and victim's cache lines are random and independent across runs, thus defeating any contention-based attack, since the attacker loses the ability to create contention for specific victim's data.

Implementing per-process unique seeds. In order to prevent contention-based attacks with hRP and ERM, a different seed has to be provided to each process which requires some Operating System (OS) support, see Figure 5.2. In the context of AUTOSAR, applications are divided into Software Components (SWCs), and each

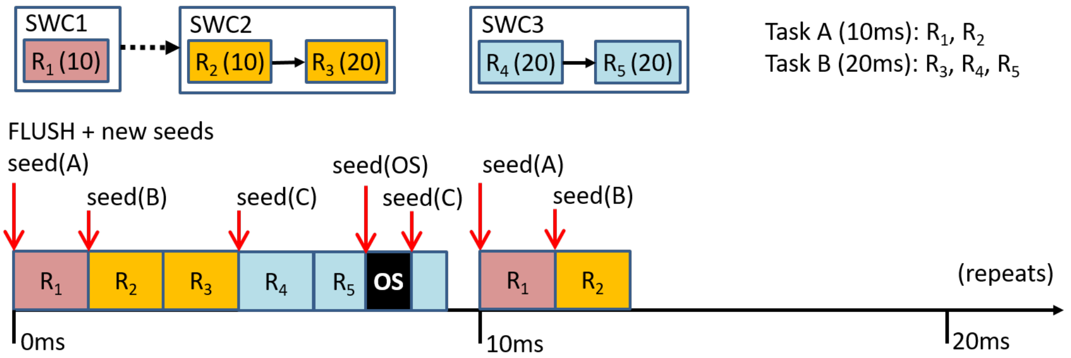


Figure 5.2: Example of AUTOSAR application and seed management.

SWC is further divided into runnables (the atomic unit of execution). Each runnable has an associated execution period, see Figure 5.2, where an application has 2 SWCs (SWC1 and SWC2), and another 1 SWC (SWC3), consisting of 1, 2, and 2 runnables respectively. For instance, runnable R_2 executes every 10 ms and it produces some output read by R_3. Also, SWC1 produces some output read by SWC2. Communication across runnables in the same SWC can be done via shared memory, whereas across SWC with message passing. Finally, runnables of different SWC are organized into tasks where each task has a specific execution period. For instance, task A includes all runnables with period 10 ms (R_1 and R_2). Runnables are scheduled within a task preserving application dependencies.

In order to allow the communication between runnables of a given SWC via shared memory, with our proposed TSCache, all runnables of a given SWC *must* use the same seed. Preserving the same seed across runnables of the same SWC also reduces the number of cache flushes and hence, overheads. In the case of runnables of different SWC, they may have been developed by different providers (even if they belong to the same application). Hence, they *must not* use the same seed to prevent contention-based attacks across SWCs[2]. This implies that, on a context switch across runnables of different SWC, the OS must store the seed in the *task struct* of the SWC, empty the pipeline, and restore the seed of the incoming SWC. This way SWCs cannot learn from the cache behavior of the other SWCs. This is indicated in Figure 5.2 with red arrows and the seed that needs to be set. Note that whenever the OS is invoked (e.g., during R_5), the OS seed needs to be used for memory coherency and to prevent also contention-based attacks. Finally, whenever the whole hyperperiod elapses (20 ms in the example), the OS needs to set new random seeds and flush cache contents. This ensures that execution times across runnables in different hyperperiods are random and independent. Note, however, that if the instances of a given runnable within a hyperperiod use the same seed, their timing is not independent (e.g., the two instances of R_1). The only practical implication is that their – arbitrarily low – exceedance probability is not independent and, if one of those instances would ever

[2] Note also that by enforcing different seeds across SWCs the system is also protected from memory attacks exploiting software vulnerabilities.

overrun its pWCET, all other instances of this runnable in the hyperperiod could also do it.

Despite seeds are changed often, cache is not flushed, so the overhead is negligible (emptying the pipeline and updating seed registers). Instead, cache flushing occurs only once per hyperperiod, as already needed for MBPTA compliance.

5.6 Case Study

5.6.1 Methodology

5.6.1.1 Berstein Attack and Threat Model

Due to the current nature of ATS, Bernstein's attack [41] is a realistic scenario, since attacker and victim do not have to share the processor during the contention attack, thus being less restrictive. Additionally the vector of attack for Bernstein's intrusion matches the processor architecture currently used in ATS (i.e., the cache hierarchy).

We emulate two independent processors that execute cryptographic operations independently, the victim and the attacker. Both processors execute 128-bit AES encryption functions. For the attacker the key is known, for the victim, a randomized 128 bits key is generated. We collect then timing measurements from the processes of encryption, and then we perform a statistical correlation on the timing profiles of attacker and victim to find the secret victim's key. In the original Bernstein's experiment, victim's timing measurements were taken on the victim's machine to reduce interference. Hence, performing the analysis on-line or off-line gives exactly the same result. For this experiment victim and attacker obtain 10^7 samples of AES encryption operations each.

We try Bernstein's attack on different setups, basically extracting for each 16-byte input value the average computation time per byte and value. In particular, we evaluate the robustness of the different setups by executing the attack and observing how much information (bits from the key) the attack is able to disclose. When computing the correlation between execution times observed and key values, we use for each byte the most stringent correlation factor so that (1) the number of combinations preserved is minimized while (2) keeping the correct value among those regarded as feasible. Hence, this is the best case for the attacker.

5.6.1.2 Experimental Setup

We use a cycle accurate simulator based on SocLib [99], modified accordingly to include the RPCache, hRP and ERM caches. The processor setup resembles the NXP e200z4 [95] a single-core automotive microcontroller chip. Details about the architecture modeled can be read in Chapter 3.

We evaluate four different setups: (a) *deterministic*: a baseline vulnerable processor with time-deterministic caches; (b) *RPCache*: a secure processor implementing the RPCache [126]; (c) *MBPTACache*: a processor implementing a random cache for MBPTA compliance; and (d) *TSCache*: our proposal to simultaneously handle

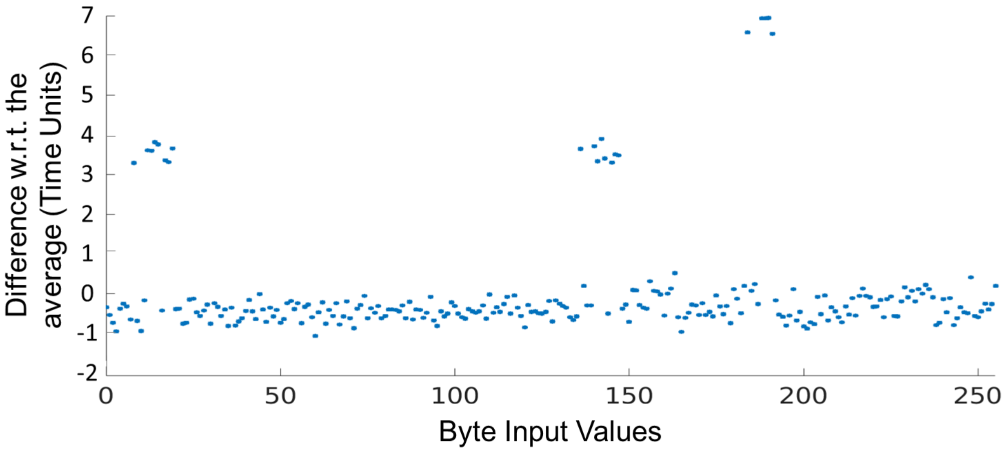


Figure 5.3: Time variations with respect to average across all different values of input byte number 4.

timing and SCA. For MBPTACache and TSCache, the L1 caches implement ERM while the shared L2 cache hRP. Further details on the experimental setup can be found in Chapter 3.

5.6.2 Results

5.6.2.1 SCA robustness

In Figure 5.3 we show how certain values for a given input (byte number 4) take slightly longer to be processed on the baseline *deterministic* setup. Those values with higher execution time allow the attacker to retrieve the value of the particular byte of the key or, at least, reduce the number of potential combinations drastically so that a brute force exploration of the (limited) remaining combinations can break the key.

Figure 5.4 shows for each cache setup the different bytes of the key in the y-axis and their potential values in the x-axis. White cells correspond to values effectively discarded by the attack, whereas grey cells correspond, to values that could not be discarded. Black cells correspond to the particular value of the key. Hence, the whiter, the more effective the attack. As shown, Bernstein's attack is effective for half of the bytes on the deterministic setup (top left plot): the attack is able to determine 33 bits out of the 128 bits of the key and other combinations are also discarded, the number of remaining combinations decreases to 2^{80}. This number of potential key values is below the 2^{128}, decreasing the cost of a brute force attack by a factor of 2^{48}.

For the RPCache, the same bytes as for the baseline setup are vulnerable to the attack. However, the RPCache proves to be stronger in front of this attack by keeping the number of potential keys to explore at 2^{108}. Still, some information is leaked. When using the MBPTACache (bottom left plot), vulnerability to the attack occurs in different bytes as for the other caches. Overall, the number of potential key values is 2^{104}, thus close to the case of the RPcache.

Finally, with the proposed TSCache, the best case attack is unable to disregard any

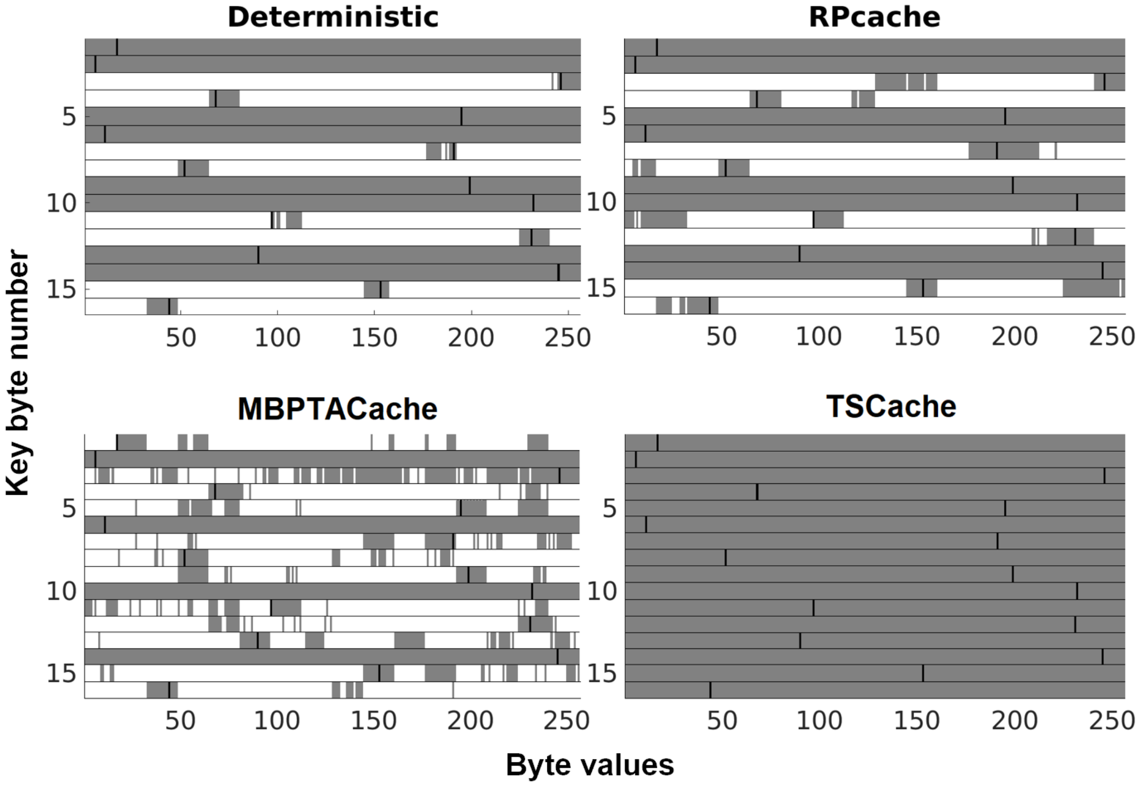


Figure 5.4: Effectiveness of the Bernstein's attack. Black cells show the used value of the secret key, grey values show attacker's considered candidates while white values show discarded values. Note how in TSCache the attacker is not able to successfully correlate the potential key values with the actual ones, and hence, no value is discarded.

value for any byte. Since TSCache makes placement fully random and independent of the actual address accessed across different seeds, Bernstein's attack fails to reveal any information, thus preserving key strength at 2^{128}. In fact, Bernstein's attack regards some values as more likely to be the key ones for several bytes, but discarding the key value for some of those bytes. Hence, TSCache, rather than preventing the attack from inferring any information by transforming it into noise, fools the attacker by providing wrong information that would not allow a brute force attack to reveal the key if fewer combinations are explored.

Generalization. Contention-based attacks, such as Bernstein's one, rely on deterministic eviction of controlled cache lines. Hence, Prime-Probe and Evict-Time Attacks, both contention-based, are thwarted by using secure time-predictable caches since the cache layouts of different processes are completely independent and randomized. As explained in Section 5.4, those attacks rely on the ability to reproduce and infer from timing profiles the inputs used by the victim. By having different seeds for victim and attacker tasks, their input state differs and so the timing profiles also differ. Hence, contention-based attacks cannot relate execution time variations with any other information, thus failing as Bernstein's one.

5.6.2.2 MBPTA-compliance

TSCache achieves Partial APOP-fixed Randomness properties (mbpta-p3), maintaining MBPTA-compliance (see Section 5.2). We further validated that the observed execution time fulfills the independence and identical distribution properties as required by EVT as used in MBPTA. We use the Ljung-Box independence test [58] to test autocorrelation for 20 different lags simultaneously, a very strong independence test. We have also applied the Kolmogorov-Smirnov two-sample i.d. test [57]. All our samples have passed both tests for a $\alpha = 0.05$ significance level.

5.6.2.3 Overheads

For the sake of clarification we discuss on the implications on hardware of the modifications needed to implement hRP and ERM in terms of performance and area.

Area. ERM and hRP caches have already been implemented on a LEON3-based multi-core processor causing no operating frequency degradation on an FPGA [37]. In terms of area, while we cannot isolate the cost of cache modifications, making the whole processor MBPTA-compliant (so modifying all caches, bus arbitration and FP units included) and adding an enhanced tracing feature costed less than 1% processor area increase.

Performance. hRP and ERM have been shown to have no effect on the maximum operating frequency of their FPGA implementation [37]. Also, they have similar cache behavior to that of standard modulo placement. Specially ERM has shown a miss rate 1% far from modulo [37], hence with negligible impact on average performance. The impact in performance due to the seed change is also negligible. Seed changes are produced for security reasons for which restoring the seed of the process to be executed next would only require to wait until all accesses in flight of the previous process have been served, which would take tens of cycles. Also changing the seed for time-predictability reasons implies flushing the cache. However this is required at coarser granularity, hence the relative cost of flushing is contained.

5.7 Other Security Implications of Applying Randomization

Following the argumentation, we note how randomization can also be applied in other regions of the processor to increase security against other security threats different from SCA. We provide a qualitative analysis of how existing solutions based on randomization could be used for protection against other security threats. In particular we reason about other types of side-channel attacks, Unauthorized Control Information Tampering (UCIT) vulnerabilities (see Section 2.4.2.1) and Denial of Service (DoS) attacks (see Section 2.4.2.2).

5.7.1 Power-Based Side-Channel Attacks

The amount of power dissipated by a program can also leak cryptographic information. When instructions execute fixed-time repetitive operations, like cryptographic algorithms that use multiple iterations for a given secret key, attackers can match power profiles obtained to infer the cryptographic data.

Randomizing the execution time to achieve protection against power analysis attacks was proposed in [130] by introducing random noise via randomly interleaving dummy instructions with the actual code when the execution of encryption algorithms is detected. However, memory layout randomization schemes such as those implemented in [35] already randomize the execution time exhibited by the processor thus being a better option to protect from both sources of attacks, namely contention-based and power analysis attacks.

5.7.2 Unauthorized Control Information Tampering (UCIT)

Randomization offers a path to address UCIT attacks by relocating both position independent and dependent regions either by software or hardware means. Different randomization schemes based on memory layout randomization can effectively protect the system from UCIT vulnerabilities by randomizing the locations of critical data elements and thus, making difficult, if at all possible, for an attacker to exactly determine the runtime locations of vulnerable points using experimentation. Coarse-grain randomization mechanism like the transparent runtime randomization [82] suffice to protect the system from UCIT vulnerabilities. While they fail to meet other properties like protection against SCA, fine-grain randomization mechanisms like [131] can provide that protection.

5.7.3 Denial of Service (DoS)

DoS attacks can be prevented by allowing a fair utilization of shared resources. This requires (1) a proper dimensioning of system resources with per-core dedicated buffers to avoid scenarios where an attacker stalls a shared resource, and (2) adequate arbitration policies. A time-randomized processor meeting these goals is presented in [35]. It ensures a fair utilization of shared resources by (1) limiting the number of in-flight requests for each core (thus limiting the maximum delay a request can suffer due to contention), and (2) implementing a random arbitration policy that accounts for the time each core uses shared resources [35]. Hence, shared resource bandwidth is fairly shared across cores regardless of their timing behavior. Moreover, such time-randomized arbitration policy is compatible with Advanced Microcontroller Bus Architecture (AMBA) protocols for communications, since AMBA adds no constraints on the particular arbitration policy used.

5.8 Related Work

Several works mitigate different cache SCA [40, 81]. In this context, cache partitioning has been proposed to solve both, contention-based SCA [132, 126] and to achieve time predictability [133]. The idea is to disable interference by isolating cache lines across different processes. However, cache partitioning severely limits the effectiveness of shared caches in multi-cores affecting both, performance and the ability to share data across threads running in different cores [134]. This relates to the difficulties to partition also all cache buffers and queues, as well as to the performance impact of reduced cache associativity per partition. Some proposals explore this option further beyond the cache, by extending isolation to other resources (e.g., miss status holding register) and putting other measures in place to tackle information leakage holistically across the processor [135].

Randomization mitigates the amount of information leaked [132, 126, 89] and has been applied to tackle contention and reused based attacks. More recent proposals also employ randomized mapping with similar hashing functions to protect caches against contention attacks [136]. Despite the increased protection, other techniques explore how to surpass the randomization barrier by efficiently finding eviction sets [137] which would enable contention attacks on randomized schemes as long as the mapping stays fixed for enough time. Works like [138, 136], based around our proposal, try to address this by using different hashing functions for each way or remapping cache blocks. However, as stated previously, the applicability of these solutions to timing analysis is inherently compromised. Overall, to the best of our knowledge this is the first work proposing hash-based randomized hardware designs for security and SCA that also address timing analysis.

5.9 Summary

Increasing performance needs in ATS requires the adoption of high-performance hardware features such as caches, that however, challenge time-predictability and make systems vulnerable to timing-based SCA. While those concerns have been addressed individually, existing solutions have not been proven compatible for both concerns. We analyzed the suitability of the solutions devised for each concern against the requirements of the other, proving that they fail to achieve both goals simultaneously. We propose Time-Predictable Secure Cache (TSCache) which effectively delivers time-predictability for MBPTA and robustness against contention-based SCA.

In this chapter, we propose and demonstrate an effective use of MBPTA time-randomized caches to deliver both, time-predictability for MBPTA and robustness against contention-based Cache-Timing Side-Channel Attacks. We assess its effectiveness against the Bernstein's attack proving that it preserves the strength of the key. We also consider the benefits of Time-Randomized Processors (TRP) against other types of security intrusions like UCIT or DoS attacks.

Part IV

Energy in CRTES

Chapter 6

Worst-Case Energy Consumption, a New Challenge

> *"If knowledge can create problems, it is not through ignorance that we can solve them."*
>
> — Isaac Asimov

6.1 Introduction

The number of (edge) devices connected to the Internet of Things (IoT) is on the rise, reaching hundreds of billions in the next years. Many devices will implement some type of critical functionality, for instance in the medical market this includes infusion pumps and implantable defibrillators. Energy awareness is mandatory in the design of IoT devices given their huge impact on worldwide energy consumption and the fact that many of them are battery powered. *Critical* IoT devices further require addressing new energy-related challenges. On the one hand, factoring in the impact of energy-solutions on device's performance, providing evidence of adherence to domain-specific safety standards. On the other hand, deriving safe Worst-Case Energy Consumption (WCEC) estimates is fundamental to ensure the system can continuously operate under a pre-established set of power/energy caps, safely delivering its critical functionality.

The rise of battery-powered and power-constrained critical devices makes energy a first-class citizen, as relevant as functional and timing requirements. At the Validation and Verification (V&V) level, evidence must be provided that power-control techniques do not jeopardize the safe operation of the device [139]. This relates to assessing the effect of those techniques on the timing of the software to prevent any overruns and providing evidence that they are triggered/deactivated in a controlled manner [140]. V&V of critical battery-powered devices also require obtaining guarantees (evidence) that with a given energy budget the device can effectively run all critical activities (tasks) due to battery or power source related constraints. This calls for methods and tools for WCEC estimation. In battery-powered devices evidence is needed to show that task runs (jobs) can execute adhering to their WCEC bound, so that the total energy consumed during operation is proven not to exceed battery capacity. Meanwhile, in power-constrained devices similar evidence is needed within smaller time frames to prove that energy consumed does not exceed power supply capabilities.

Intuitively, the properties required on WCEC estimates are comparable to those for Worst-Case Execution Time (WCET) estimates, namely providing tight upper bounds to actual energy consumption and evidence for certification. However, as we show in this chapter, despite the similarities in the concept, WCET and WCEC estimation are different processes subject to fundamentally different sets of requirements coming from the hardware. The latter shapes the set of assumptions that can be made on the hardware information required for tight energy measuring and modeling.

In this line, we analyze for the first time the impact that different hardware physical parameters have on both model-based and measurement-based WCEC modeling, for which we also show the main challenges they face compared to chip manufacturers' current practice for energy modeling and validation. Under the set of constraints that emanate from how certain physical parameters can be actually modeled (see Section 2.4.3), we show that measurement-based WCEC is a promising way forward for WCEC estimation.

6.2 Sources of Power Variability

Two are the main physical factors that particularly complicate power estimation at the hardware component level.

1. The power dissipation of any hardware component (e.g., the whole processors or a floating point unit) varies across units[1]. Furthermore, power dissipation figures differ from their (theoretical) nominal value. This relates to physical variability for hardware manufacturing.
2. The power dissipation of a given component varies over time in each unit due to several sources of variation.

Operation-time (fabrication) Process, Aging, Voltage and Temperature (PAVT) variations cause that, even if hardware designers could model circuits at the lowest (most-accurate) level, designers would still miss the actual variations experienced by each individual processor unit. This seriously complicates – in fact makes it de facto impossible – predicting exactly power consumption a priori. Furthermore, specific processor unit(s) under study are used to derive power estimates for all of them.

Process Variations. Limitations in the manufacturing process cause device (e.g., wires, vias and transistor components) parameters (e.g., geometry, thickness and number of dopants) to differ from their nominal values. Taking as an example the lithographic process, variations have a systematic and a random component. The

[1] A unit is a physical implementation of a given component (e.g., a processor may have two floating-point units; also two single-core chips are two different processor units).

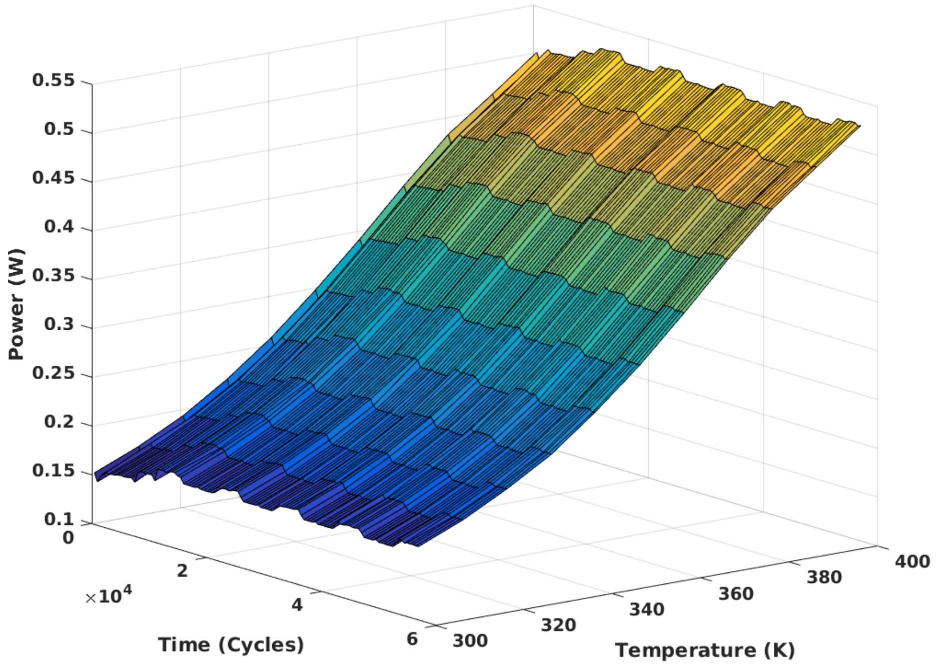


Figure 6.1: Average power dissipation of a program through execution time for different temperatures. The binary alternates execution of memory and FPU instructions on an in-order 4 stage processor with separate instruction and data level 1 caches, and a unified level 2 cache.

former manifests in spatial correlation so that variations affect in a similar manner neighboring devices; while the random component refers to individual devices suffering independent variations. Variations make delay and power dissipation of each individual device differ from nominal values and at a coarser granularity, those variations lead to delay and power variability of processor components. For instance, 3x power variations with 90 nm technology [141] and 20x leakage (static) power variations [142] with 180 nm technology have been reported by industry across different processor units (between the most power efficient and the most power hungry units).

Accounting for those variations requires studying each different processor unit separately, or using statistical means [143] to determine average or maximum power dissipation.

Aging Variations. Electromigration, Bias Transistor Instability (BTI) and Hot-Carrier Injection (HCI) [144, 145, 146] (among others), affect the resistance of wires and threshold voltage (V_t) of transistors. They also change processor behavior (including energy consumption) over time and affect physical characteristics of the devices by displacing molecules and dopants from their original locations. Hence, power dissipation for a unit slowly changes over time.

Temperature and Voltage Variations. Processors operate within a given temperature and supply voltage range. Both of them vary due to the activity of the whole processor, ambient temperature and physical characteristics of the supply source, package, processor pins, etc. For instance, if some cores in a multi-core move from idle to active, they will increase switching activity, thus consuming more power. This will produce higher temperature, that will propagate to the neighbor cores, and will reduce the amount of current available for other cores, which will perceive a V_{cc} decrease. This, ultimately, affects power dissipation dynamically at very fine grain (e.g., voltage variations may occur at the scale of few nanoseconds). As an illustrative example, Figure 6.1 shows average power measurements of 500-cycle intervals for a program execution in a relevant temperature range for many embedded microcontrollers [147]. A temperature increase of 100 degrees leads to a power increase of up to 3.5x.

6.3 Current Practice on Processor-Level Typical and Maximum Power Estimation

As an initial step to define a method to derive reliable WCEC bounds, we describe current practice for low-level processor energy modeling. Arguably, chip vendors have the most advanced techniques and tools for that purpose. Hence, understanding the limitations of those models is fundamental to understand the limits of WCEC estimation. Note that chip vendors are interested in determining suitable cooling solutions, so their focus is on sustained power estimation under highly stressful scenarios.

Power models and measurements are used to estimate power during processor design [148, 149]. They help iteratively modifying the design until there is enough evidence that target peak power values are not exceeded (see Figure 6.2). During the process, chip vendors also use techniques such as adaptive body bias[2] [150] to trade off between maximum operating frequency and power dissipation of the processor. Due to the known inaccuracy of the models at the different abstraction levels, safety margins are applied to account for the unknown, such as deviations in the actual switching activity estimated, the impact of PAVT variations or the effectiveness of the cooling solutions [151].

Models. Model-based techniques are known for being slow, limiting the *window of analysis* to few thousands of cycles at most. For instance, in electrical-level SPICE models, characterizing a memory macrocell with synthetic stimuli can take days of simulation, with a single BSIM4 CMOS transistor model accounting for more than 40 parameters [152]. On the one hand, the huge time requirements of models are handled by abstracting physical behavior keeping the model usable but reducing its accuracy. On the other hand, despite the complexity of the models, their accuracy with respect to reality may not be sufficiently high and, moreover, it is also hard to be estimated. This emanates from the limitations of the model to capture all physical effects and its inability to model *exactly* PAVT variations, often accounted for statistically [143].

Power models are used in chip industry for pre-silicon validation and design refinement (Figure 6.2), for instance for determining whether the power supply is enough, the appearance of power hot-spots and the efficiency of cooling solutions. Models

[2] Body bias techniques rely on modifying the voltage of the substrate to either increase threshold voltage (V_t) so that leakage power and speed decrease; or to decrease V_t causing an increase in speed and leakage power.

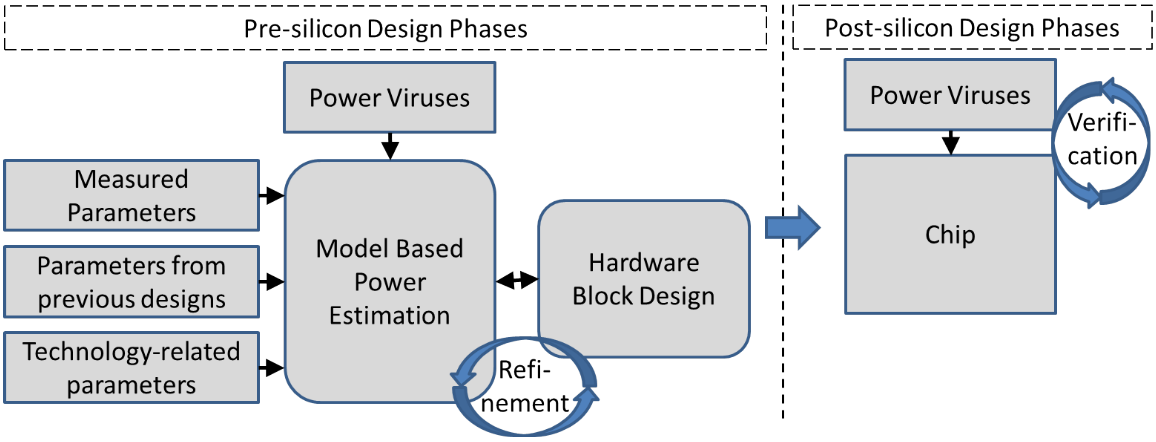

Pre-silicon Design Phases
Post-silicon Design Phases
Power Viruses
Measured Parameters
Parameters from previous designs
Technology-related parameters
Model Based Power Estimation
Hardware Block Design
Refi-nement
Power Viruses
Verifi-cation
Chip

Figure 6.2: Usage of (measurement/analytical) models and measurements during the hardware design process.

comprise an analytical part and a wide set of parameters obtained from measurements on 'prototype implementations' such as macrocells, small prototype chips, etc. or technology projections derived from previous implementations on similar technology (feature size) [153]. The model is evaluated on small hand-made kernels (power viruses) to derive extreme behavior. However, power viruses do not guarantee that the worst power is captured. This relates to the difficulties to produce those inputs leading to the worst switching across the full chip, under the worst PAVT variations conditions. Identifying the sequence of inputs needed for each Functional Unit Block (FUB) of the processor is simply unaffordable. Then, producing those inputs *simultaneously* in all FUBs is more challenging requiring controllability to produce the worst combined inputs and preventing to control PAVT variations.

Measurements. Measuring actual power consumption in real processors is limited by the availability of power monitoring units. The granularity at which power readings can be provided is coarse in time (e.g., 1 second [154]) and space (e.g., components in the pipeline can neither be isolated nor accessed physically to measure their power dissipation). As a result, engineers stick to external means to take coarse-grain power measurements. Interestingly, while some processors provide built-in power monitors for some components, those are power-proxy approaches with which power is derived as a linear model of performance monitoring counters (activities), which are weighted by constants. Those constants are derived empirically with a regression model from the execution of several reference applications. This is the case of the IBM POWER7 [155].

Measurements are used for post-silicon validation (see Figure 6.2). Due to the complexity of achieving accurate power estimates analytically, chip vendors verify chip power using actual measurements – despite their own limitations. This allows deriving power and energy figures for the different processor components. The main challenge for deriving worst-case energy and power measurements resides on the definition of representative scenarios. For example, maximum peak power numbers for processors are obtained using benchmarks that generate the most (*expected*) stressing situations, aka, power viruses [156].

Despite advanced models and measurement approaches, the risk of inaccuracies is not removed. One of the most well-known failures in the prediction of the peak/typical power, is the Intel Tejas processor (aka, Pentium V), which finally exceeded its power/temperature budgets due to model inaccuracy at a level that even body biasing could not correct, so its production was abandoned [157]. Although these practices are costly and not always effective, they are still affordable and used in practice by experts due to being the most accurate methods available.

Summary. Overall, model-based approaches build on detailed knowledge of the system. The applicability of this type of white-box approaches is challenged by the lack of details of real processors. In contrast, measurement-based approaches, a form of black-box approach, can still derive estimates through experimentation although uncertainty may remain due to the difficulties to create representative tests.

6.3.1 Validation and Verification

Criticality derives from functional safety and safety standards (e.g., IEC 62304 for medical devices and IEC 61508 for industry). Interestingly, safety standards do not aim at removing the appearance of failures, which is arguably impossible in a real system. Instead, they aim at making their likelihood of occurrence to be quantified and assessed against reference values, asserting with sufficiently high confidence that the residual risk of violation falls below tolerable rates. In this line, despite common wisdom, systems are designed such that a task overrun never leads to an unsafe state of the system, which would mean a bad-designed safety solution. A safety process is defined (according to the corresponding standard) covering the definition of safety goals and requirements, and a safety strategy in general, to mitigate the risk that hardware or software misbehavior causes a system failure. As the criticality of the software component under analysis increases, more mechanisms are put in place (replication, online monitoring, watchdog) to detect and react to undesired situations.

6.4 Model-Based Task-Level WCEC

In this section, we detail the main difficulties found by static (model) based approaches to estimate the maximum energy consumption of a given task. Intuitive solutions based on multiplying the average power consumption and the WCET estimate for a task may not lead to high-quality WCEC estimates since energy and time do not necessarily correlate [158].

Few works address the problem of WCEC estimation from an analytical point of view [158, 159]. Following the principles of static WCET analysis, model-based (static) WCEC analysis builds on deriving a cost function for each instruction, with the (obvious) observation that the latter uses energy as cost function. Energy cost is derived at instruction level and then combined to derive energy cost of basic blocks.

From that point on, standard Integer Linear Programming (ILP) formulation – or any other sound formulation – is used to derive WCEC estimates for the task.

WCEC techniques work at a high abstraction level compared to what we discussed in the previous section. Those techniques focus on pipeline effects (Fetch, Decode, etc.) and hardware components used in each stage (e.g., caches, functional units and the like). As reference figures to compare against, WCEC models use estimates provided by open-source power models. Those models are generic (i.e., not tailored to any particular processor) and can indistinctly result in over- or under-estimates. Hence, obtaining WCEC estimates above the estimates provided by the reference model does not guarantee high-quality WCEC estimates as they can be lower or far higher than the actual energy figures.

6.4.1 Granularity and Accuracy

There are several levels at which power can be modeled, such as (in increasing order of abstraction) electrical (e.g., SPICE models), gate level and Register-Transfer Level (RTL). At the highest levels, small programs are used to derive power estimates for a given hardware component. These programs are usually restricted to small power viruses [156] that aim at generating high power consumption by, for instance, increasing the activity factor.

Modeling full-program energy consumption poses many challenges. One of them relates to keeping the execution time requirements affordable, which inevitably results in simplifying the underlying power model. In particular, the number of physical details factored in is reduced, which basically plays against the accuracy of the power estimates. Model simplifications may cause inaccuracies either under- or overestimating power. For instance, gate-level or RTL models lose some accuracy and can only be afforded to simulate small programs (e.g., simulating a full processor during several thousands of execution cycles may require several days of simulation). As the complexity of the models decreases to make the problem tractable, information such as the switching activity of the transistors is lost. A feasible approach to increase the granularity minimizing the impact on accuracy would be using measurements coupled with statistical bounding analysis at the desired granularity level as inputs for the models. Following this approach, any implementation-dependent factor is captured by the measurements and upper-bounded by statistical formulation. For instance, the specific implementation of an adder (e.g., carry-lookahead and Kogge-Stone) determines how its transistors switch, and the physical implementation determines the size of the different transistors, so how much capacity switches for each transistor. Hence, the lack of accurate information on the capacity switched on each operation in each component may lead to large inaccuracies.

6.4.2 Upper-Bounding the Activity Factor

The activity factor (aka, switching activity) of a given FUB is a figure in the range 0-1 that describes the fraction of the total capacity of the FUB that switches (and hence consumes dynamic power) in a particular processor cycle. The activity factor plays a key role when estimating dynamic power (see Section 2.4.3). Deriving the activity factor for a FUB requires extensive knowledge about the particular transistors (and their geometries) whose inputs change on a FUB input change, if not every cycle, on average.

First, many processor details are not visible at the software level. For instance, it is inconceivable to devise how control signals switch (e.g., to manage queues between pipeline stages) from the abstract analysis of program instructions.

Second, this information can only be obtained with transistor-level simulations, which incur huge overheads to enable modeling full programs. Note that chip vendors may not make those details public for competitive reasons. Additionally, it is simply unaffordable precomputing the energy consumption of all potential input transitions due to computational and storage cost. As an illustrative example, let us assume a particular FUB such as an adder. A 32-bit adder has, at least, 2^{64} different inputs if we ignore control signals, and so there are at least 2^{128} different input transitions possible, each one producing a specific capacity to switch. Identifying the worst possible transition analytically (out of the thousands of transistors) or empirically is beyond the reach of any circuit designer which, at most, can guess what the worst transition is. Hence, the complexity of obtaining and managing such detailed information is beyond the reach of static models.

An intuitive way to handle this, as done in WCET analysis, is making pessimistic assumptions. For instance, switching activity is assumed to be 1 since providing evidence that a lower value is an actual upper bound would resort to unaffordable low-level information/models. However, typical switching activity is largely below 1 due to idle blocks whose inputs do not change in specific cycles, or due to the usual bias of input values operated and stored towards specific values, which lead to very limited switching activity. To provide concrete empirical evidence on this general intuition, we show an example that builds on the so called *toggle factor* in Table 6.1. It represents the fraction of nodes[3] that have switched at least once in the processor and hence, can be regarded as an upper-bound of the switching activity of a circuit since only a subset of the transistors in the toggled nodes have effectively switched. In particular, we have computed with the QuestaSim RTL simulator the toggle activity factor for several benchmarks executed in an RTL LEON3 processor description. As shown, only around 40% of the nodes toggled (i.e., the activity factor is at most 40%, but typically much lower).

Worst-case assumptions on the activity factor result in remarkably pessimistic estimates. On the previous example, and assuming that half of the transistors switch in a toggled node, the processor could consume 20 W, while we would account for 40 W assuming the toggle factor, and 100 W assuming switching activity 1. Hence, the estimated WCEC may implicitly lead to a power dissipation above the actual capabilities of the processor, reducing its practical use. As explained before, switching activity decreases exponentially (often quadratically) across gate levels, so activity factors of up to 5% are expected for simple circuits [160]. Lower factors are expected for more complex circuits.

[3] A node in RTL represents a high number of transistors in the actual circuit.

Table 6.1: Toggle coverage for different Workloads on a RTL model of the LEON3.

	EEMBC AutoBench			Mälardalen WCET	
IU **components**	rspeed	canrdr	ttsprk	matmult	firFn
Fetch	72.58%	72.58%	74.19%	57.26%	58.06%
Decode	70.27%	68.92%	72.30%	63.51%	60.13%
Register access	80.00%	77.88%	79.70%	73.33%	71.82%
Execute	77.78%	76.19%	77.38%	73.54%	72.22%
Memory access	62.43%	60.22%	62.15%	65.19%	74.86%
Exception	66.51%	64.22%	65.82%	76.15%	76.26%
Write back	29.19%	28.57%	28.57%	26.09%	45.96%
Data cache	57.52%	57.21%	57.52%	56.44%	57.06%
Instruction cache	41.32%	41.32%	42.36%	35.33%	41.32%
Register file	92.48%	92.48%	92.48%	87.97%	97.97%
Others	12.38%	10.6%	10.78%	15.32%	16.09%
Total	39.2%	37.8%	38.5%	39.8%	41.6%

6.4.3 PAVT Variations

As detailed in Section 6.2, PAVT variations can produce large power variations across units. Any static WCEC estimation model aiming at providing arguably sound energy upper-bounds – that cannot be exceeded under any circumstance – cannot afford using typical values or values obtained from statistical distributions (e.g., mean plus six sigma). The latter can be probabilistically exceeded and, even if that could occur with a negligible probability, it cannot be proven to be zero. Such a WCEC estimation approach confronts with chip vendors' current practice: simply deriving the worst possible value is out of the reach of chip manufacturers that, instead of relying on a theoretical value, build upon measurements to determine the parameters of a Gaussian distribution matching best the observed values. Then, an upper-bound value is chosen based on N-sigma approaches. In other words, industry resorts to measurements to determine bounds to different parameters and use as upper-bound the mean (μ) plus N times the standard deviation (σ), where N is typically in the range 3-6, depending on the exceedance rate that can be afforded for that particular component and metric [161].

Interestingly, even if we assume that the highest observed value is a true upper-bound, in practice due to the uncertainty brought by test campaigns on specific processor units, the degree of pessimism for power estimation can be huge. For instance, process variations may produce power discrepancies of 3x across processor units [141], voltage variations can produce ≈25% power variations [142], and temperature variations around 3.5x power variations as shown before. Therefore, even neglecting aging variations, PAVT variations in power (and so in energy) can be as significant as 13x if the absolute worst case needs to be accounted for.

6.5 Measurement-based WCEC Estimation

To our knowledge, no measurement-based WCEC estimation technique exists. Next, we detail the main aspects of WCEC estimation for tasks with measurement-based approaches.

6.5.1 Quality of the Measurements

Using the target platform for collecting power measurements offers the advantage of speed and removes discrepancies with reality due to modeling. Furthermore, measurement-based analysis can also handle complex scenarios by mimicking real-world workloads (i.e., multiple tasks running simultaneously) through the use of stressing tests and operation conditions (e.g., high temperature), thus accounting for interactions between tasks by merely executing them together without the need for any detailed model (i.e., a form of black-box approach). Moreover, a measurement based technique can capture the effects of multi-task workloads and their interactions between them. Whenever some effects cannot be properly accounted for through measurements, then disabling or enabling some features (e.g., cache partitioning) can limit the complexity of multi-task workload interference. The other side of the coin are the challenges to observe and account for PAVT variations as well as software-dependent (internal) effects.

Regarding observability, while power meters can be used, they may create some effects on the power consumption of the processor due to the coupling of the power supply lines and may have some degree of inaccuracy. Moreover, power meters measure the power of the full processor rather than the power of the task only, so deducing task energy consumption can only be done with separate experiments running and not running the task, but some non-controlled PAVT variations may interfere measurements differently across runs. Finally, synchronizing the start of the run of the task with measurement collection is a tough task, so measurements need to be collected at a coarse granularity (e.g., for 1,000 runs of the task with identical inputs) to mitigate this effect.

Regarding PAVT variations, process variations correspond to those of the actual processor being used, so how they represent other processor units can only be studied statistically using other processor units, extrapolating the effect from small-scale experiments on simulated platforms or with data provided by the manufacturer. Analogously, aging, voltage and temperature conditions observed may be representative of neither the typical case nor the worst case. Thus, it may be required to use simulations for extrapolating their typical effect for statistically relevant scenarios.

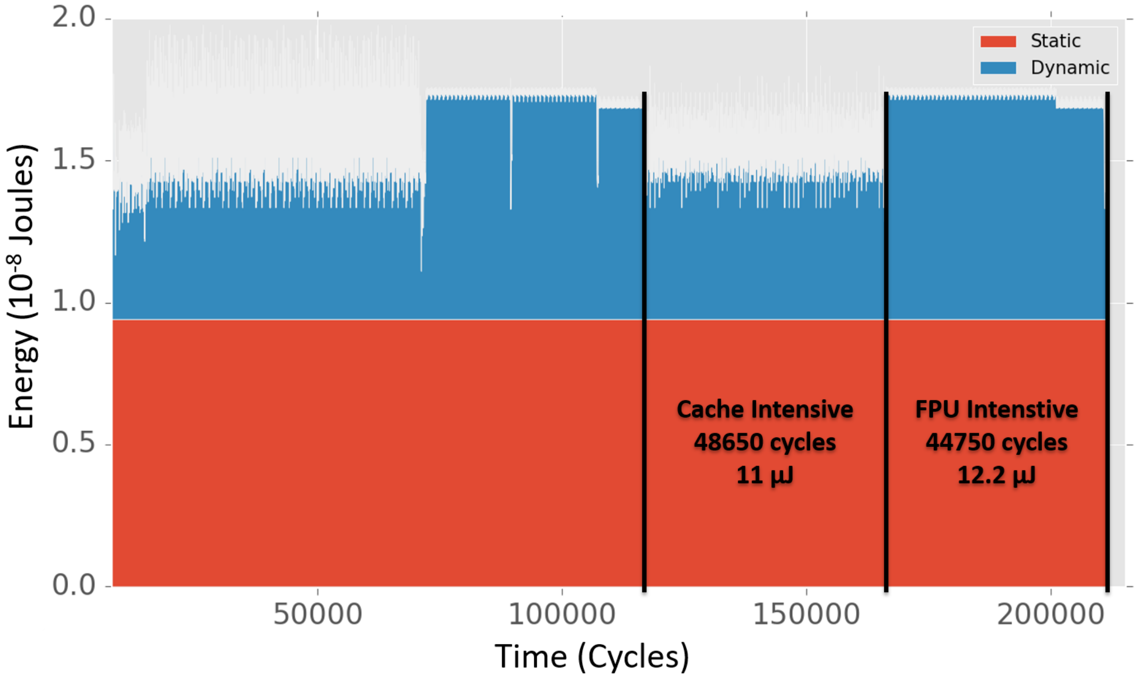


Figure 6.3: Average energy consumption of a two-path program. Note that, even though the cache intensive path takes more cycles, the FPU intensive path consumes more energy.

6.5.2 Input Space Coverage and Representativeness

As for timing analysis, measurement-based WCEC analysis has to deal with all challenges related to input-space coverage such as program path coverage and memory placement of objects (and its influence on cache behavior), both in single-core and multi-core execution environments. However, differently to timing analysis, some of these factors have non-obvious effects on power and, moreover, a number of parameters may be innocuous for performance, but not for power.

Cache behavior correlates well with energy in general, with hits served faster and with lower energy than misses. The latter need to further access another cache level or memory and take longer to be served. Still, it is possible finding specific examples where hits lead to higher energy consumption than misses.

Execution Paths. While path coverage is equally important for both, timing and energy analysis, the challenge for energy relates to the fact that higher execution time does not imply necessarily higher energy consumption. First, there is a direct relation between execution time and energy consumption due to static energy, which is roughly proportional to execution time. Thus, in general, paths with longer execution time will likely produce higher energy consumption, but only if the instruction mix and values operated are similar enough. And second, an execution path that incurs many cache misses may take longer than a computation intensive path. However, the latter may produce much higher switching activity due to computation than the former, where the pipeline stays mostly idle with low switching activity. Figure 6.3 illustrates

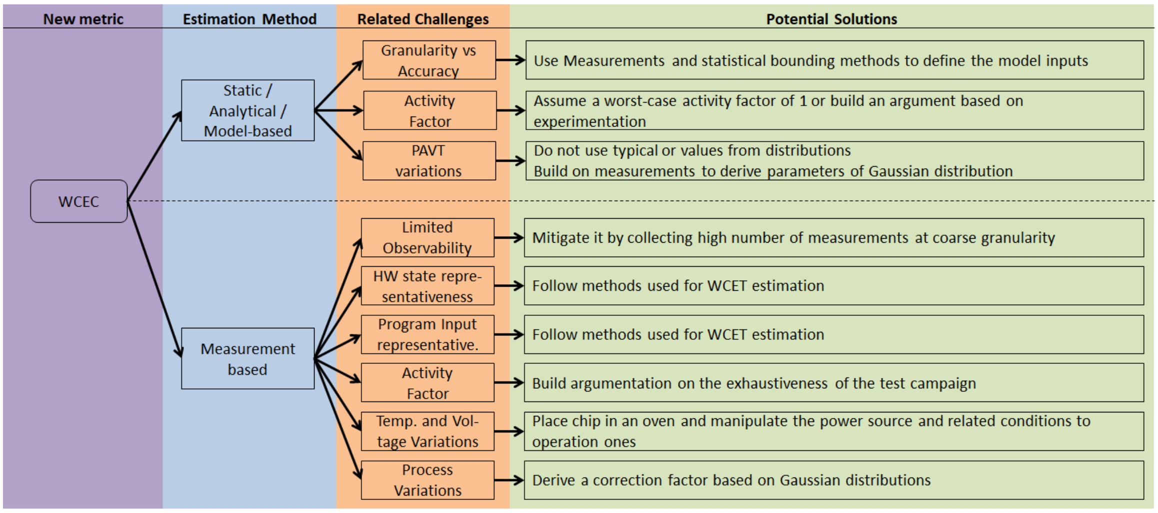

New metric
Estimation Method
Related Challenges
Potential Solutions
WCEC
Static / Analytical / Model-based
Granularity vs Accuracy
Activity Factor
PAVT variations
Use Measurements and statistical bounding methods to define the model inputs
Assume a worst-case activity factor of 1 or build an argument based on experimentation
Do not use typical or values from distributions
Build on measurements to derive parameters of Gaussian distribution
Measurement based
Limited Observability
HW state representativeness
Program Input representative.
Activity Factor
Temp. and Voltage Variations
Process Variations
Mitigate it by collecting high number of measurements at coarse granularity
Follow methods used for WCET estimation
Follow methods used for WCET estimation
Build argumentation on the exhaustiveness of the test campaign
Place chip in an oven and manipulate the power source and related conditions to operation ones
Derive a correction factor based on Gaussian distributions

Figure 6.4: Diagram of the main challenges, and potential paths to follow, addressed by model-based and measurement-based WCEC estimation.

this effect by showing dynamic and static energy consumption for 50 execution cycles intervals for a multipath program with 2 paths, each executed twice. On the second iteration of the program, the first cache-intensive path takes 48650 cycles to execute and consumes 11 μJ while the second computation-intensive path consumes more energy (i.e., 12.2 μJ) and has a shorter duration (i.e., 44750 cycles).

Overall, assessing the relationship between energy and execution time for a given task, or simply identifying the paths leading to the highest energy consumption for a task, is an open challenge. Initial solutions can build on those derived for WCET based on using the input data used for functional testing or some type of randomization to automatically cover the design space and derive probabilistic coverage arguments [49, 80].

Activity Factor. The relationship between the activity factor and input data is extremely hard to establish. As indicated before, input values for FUBs may produce high or low switching activity. This often relates to the number of changing bits across operated values, since changing bits may induce some switching activity. However, other effects such as memory placement (and so cache placement), even if performance remains the same, may lead to significantly different switching activities. For instance, different addresses may produce different switching activity when operated to add an offset. Analogously, if two addresses are mapped to the same cache set, even if their accesses produce the same hit/miss sequences, may cause different switching activity in the cache decoders, in the replacement information of the cache sets, etc. Hence, determining a realistic and tight upper-bound to the switching activity of the task under analysis is difficult. Since it depends on highly distant layers (i.e., input data for the task and transistor-level implementation of the processor), no practical means can be realistically set up to get measurable confidence. Instead, only argumentation based on exhaustiveness of test campaigns can be used, whose reliability is difficult to assess.

6.5.3 PAVT Variations

Some variations, such as temperature and voltage, can be induced during analysis by placing the chip in an oven and manipulating the power source of the processor. Yet, relating those conditions with worst-case operation conditions is a complex challenge.

Other variations, such as aging, can be accounted for applying accelerated aging on a processor. This is typically done by applying overly high temperatures and voltages so that the accumulated aging occurred in several years of operation is produced in a few hours. However, whether accelerated aging produces *exactly* the same effects as aging during operation due to physical implications of using different stress conditions is unclear.

Finally, process variations change across processor units, thus making energy estimates obtained for a given chip unit be invalid for any other chip unit. Thus, the only reasonable way to account reliably for the effect of process variations is performing the analysis on the chip to be deployed. This, however, poses a serious issue for many industries: power analysis needs to be repeated for *all* processor units delivered. This is virtually unaffordable for many industries where the number of units can be in the range of millions and cost constraints are severe. Although industry performs a number of verification tests in all units deployed to detect obvious defects, the full validation and verification process followed for certification/qualification purposes is not repeated for each system unit, including all its components. Thus, process variations also bring uncertainty to WCEC estimates. Similar to the model-based approaches, and as stated in Section 6.4, process variation effects can be accounted by analyzing a representative large enough number of processor units and obtaining its corresponding statistical and probabilistic distributions.

6.6 Putting it All Together

Power models may produce power estimates with arbitrarily large under- or over-estimate inaccuracies with respect to nominal power dissipation, which in turn, may also have large deviations with respect to the actual power dissipation of a given processor unit in a given time interval due to PAVT variations.

Complex power models have been used for low-level hardware energy modeling (e.g., hardware component, transistor, and capacitor level). Extending these models to derive WCEC estimates at the task level faces the challenges of granularity and precision, see the top part of Figure 6.4. The former covers the infeasibility of using existing slow models to scale to the size of tasks. The latter covers the fact that abstractions are needed to reduce performance requirements, which naturally cause trading some precision of the models. This translates into making worst-case assumptions for many parameters, resulting in pessimistic estimates that limit their usability and restrict them to early design stages when the objective is to derive initial tasks' energy/power budgets and task schedules that fit a given energy/power budget.

Measurement-based approaches offer proxies close enough to reality to be usable and to be understood by end users in their certification arguments about tasks worst-case energy consumption. Yet it is required to deal with several sources of uncertainty with qualitative reasoning and statistical methods as the only approaches available to ascertain the degree of uncertainty, see the bottom part of Figure 6.4. In particular, mechanisms need to be devised to mitigate uncertainty and increase confidence and representativeness of measurements collected: i) for increasing the observability of hardware and software interactions measurements have to comprise a very high number of observations first with identical inputs and later varying inputs to achieve sufficient coverage. Tests have to be intended to enable out-of-normality cases to surface to the observer; ii) for hardware-state and program-input effects, we can build on existing solutions used for WCET either based on randomization as a way to naturally explore complex interactions of software and hardware [49] or techniques based on the user's ability to build test campaigns able to cover the worst possible situations [80]; iii) the activity factor case builds upon exhaustive tests and a necessary qualitative argumentation to reliably trust those tests; iv) temperature and voltage variations can be accounted by stressing the hardware under analysis by subjecting it to extreme cases of voltage variations or applying accelerated aging; v) finally, process variation uncertainty can be reduced by the use of a large enough test pool of processor units from which to derive statistical distributions that allow the application of a correction factor on the WCEC measurements.

6.7 Related Work

Powerful tools exist to measure power at the electrical level, such as SPICE [162], or higher granularities, such as CACTI [163], which models resistances and capacitances of memory structures, and McPAT [101] and WATTCH [164], which estimate the power of full processors building upon CACTI. Literature on power and energy estimation mostly focuses on empirical regression models, dealing with the selection of the features that should be used to most effectively model energy for different types of platforms or processors (e.g., CPU, GPU and ARM Based) building upon their performance counters [165, 166, 167]. Hybrid models, which combine analytical and empirical models, are also proposed to trade accuracy for microarchitecture independence [168]. Other works approach energy modeling from a probabilistic view by using stochastic models and random distributions [169]. The use of manufacturer-provided models (e.g., Intel's RAPL) has also been considered and enhanced by several works as an out-of-the-shelf viable accurate alternative[170]. However, none of those tools or models is intended for WCEC estimation of full tasks, as needed in the context of critical real-time systems.

Other works [171] have also verified that variations across identical instances of the same processor are not negligible, which directly impacts empirical models and how to account for the worst-case across a processor pool.

However, for critical tasks and real-time systems where tight bounds on resource consumption must be defined (either time or energy) this is rather a new field.

Some authors have assessed the strong dependence between WCEC and input values of different components [172]. Others [173] assess the validity of current WCEC methods, showing that WCEC cannot be estimated with mathematical proofs, instead of requiring a shift towards a more statistical framework. In other domains, circuit high power has been predicted using Monte-Carlo approaches and Extreme Value Theory (EVT) [174] focusing on Thermal Design Point (TDP) rather than WCEC. Representativeness of the estimates is only discussed to some extent as they rely on the ability of the user to define representative testing scenarios, thus facing the same problems of state-of-the-art power verification approaches.

For model-based WCEC techniques, [158] shows that multiplying average power by the WCET is not reliable, so they build upon model-based WCEC estimates for basic blocks extrapolated from micro-architectural level power models and use Implicit Path Enumeration Technique (IPET) to estimate the global WCEC. This approach has been improved [159, 175] combining IPET with genetic algorithms to trade off between reliability and tightness and provide hard and soft WCEC estimates. However, these models work at a high abstraction level, assuming a fixed amount of energy per instruction, which fails to reflect the underlying variability at hardware level caused by Process Variation (PV) and other effects. On the other hand, it is unclear how static models could account for stochastic effects such as those of PV.

We attack the WCEC estimation problem from a different angle. Starting from current industrial practice, we identify the key elements challenging industrially-viable WCEC estimation and provide the basis for a measurement-based probabilistic approach.

6.8 Summary

Energy is a key metric in critical battery-power and power-constrainted edge devices, calling for effective means for WCEC estimation. While the theory behind timing (WCET) analysis has been developed during years, WCEC estimation has received much less attention.

In this chapter, we describe key aspects of WCEC estimation (impact of switching activity and PAVT variations) so far ignored by previous methods. To our knowledge, no previous work covers the increasing gap between WCEC estimation methods and how energy varies in real systems. We make a first step in that direction by bringing together knowledge from industrial practice on energy estimation and WCEC estimation in the embedded domain. Overall, this chapter settles the ground on the grand challenges (and directions to address them) for practical and reliable WCEC estimation and aims at becoming a reference for future works on WCEC estimation.

Chapter 7

Worst-Case Energy Consumption Modeling Methodology under the Presence of Process Variations

"*Aut viam inveniam aut faciam.*"

— Hannibal Barca

7.1 Introduction

As explained in the previous chapter, processor energy and power can suffer significant variation across different units due to Process Variation (PV) (i.e., variability in the electrical properties of transistors and wires due to imperfect manufacturing) which challenges existing Worst-Case Energy Consumption (WCEC) estimation methods for applications.

Therefore, WCEC estimation must (1) scale to arbitrarily complex software-hardware systems and (2) account for the impact of PV intrinsic to highly-integrated process technologies. PV is an inherent consequence of the processor's manufacturing process and makes transistors and wires that were initially designed to be identical, end up having significantly different electrical properties. As a result, energy consumption varies significantly across different instances of the same processor. This challenges WCEC estimation since the WCEC estimates obtained for a given chip unit are not valid for other chip units. Performing Validation and Verification (V&V) activities on every deployed chip poses a serious issue for autonomous systems industry, because the number of units can be in the range of millions and the costs are simply unaffordable (e.g., due to the low cost of drones and high chip count in cars). Although industry carries out several tests to all deployed units, the full V&V process followed for certification is not repeated for each system unit. In this context, this chapter proposes a statistical modeling approach to capture PV impact on applications energy and a methodology to compute their WCEC capturing PV, as required to deploy portable critical devices.

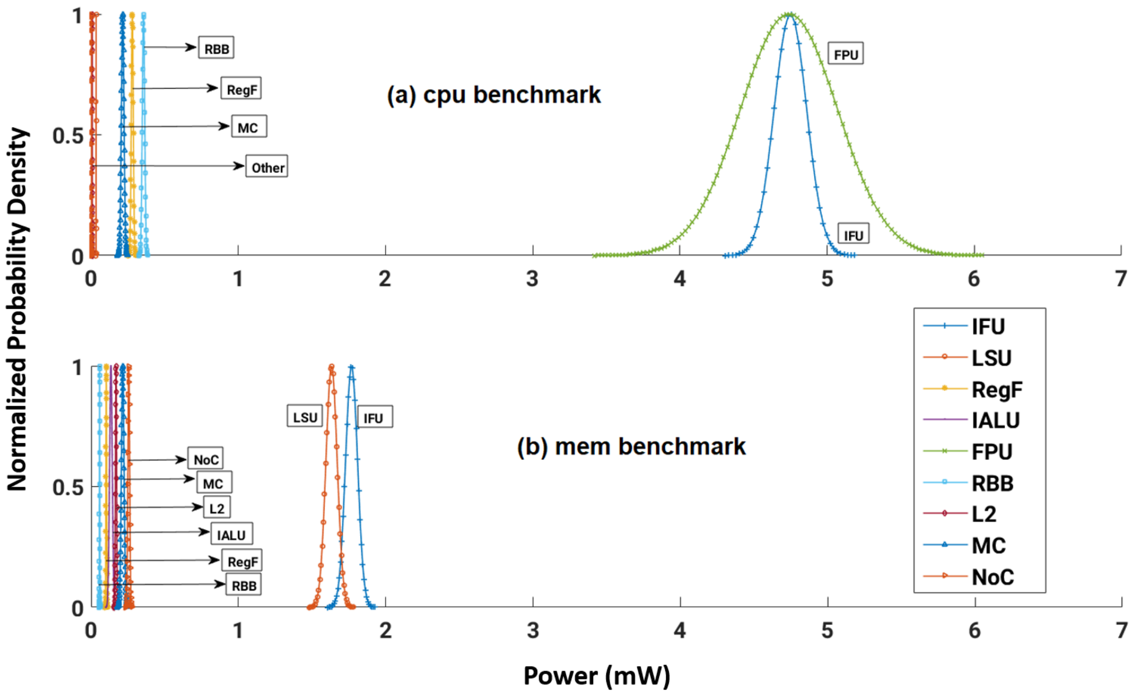


Figure 7.1: Per-FUB power variability for the CPU and memory intensive benchmarks.

7.2 PV-related power variability

PV causes transistors and wires, which are designed to have a common nominal behavior, to have different electrical properties. The chemical metal planarization process of wires creates capacitance and resistance variations that arise as a consequence of variations in wire dimensions and defects introduced in the layout [176]. For transistors, the sources of variation are systematic and random PV. Systematic PV causes the effective length of transistors to vary as a consequence of imperfections in the photo-lithographic process. The most important source of random PV is random dopant fluctuations.

PV makes power consumption vary across different processor units (instances) and Functional Unit Blocks (FUBs) (i.e., instances of functional units on the same processor unit). The latter is due to the fact that different FUBs are built using different transistors and wires with radically different electrical properties. For instance, memory structures included in processors are built using full custom transistor-based bit cells (6T, 8T), whereas simple combinatorial logic is built using logic gates created with standard-cell libraries.

Within-chip PV, while less severe than chip-to-chip PV, makes the effects of manufacturing deviations be different across FUBs. For instance, within-chip systematic PV, caused by imperfections in the photo-lithographic process, presents a strong spatial correlation causing distant transistors to present different manufacturing deviations.

Specific per-FUB PV creates an indirect dependence between the specific software

FUB	Leakage	Mem. bench.	CPU bench.
Instruction Fetch Unit (IFU)	31%	29%	26%
Load/Store Unit (LSU)	31%	30%	31%
Register File	31%	24%	18%
Integer ALU	28%	16%	28%
FPU	40%	40%	18%
Result Broadcast Bus (RBB)	17%	17%	14%
L2	31%	31%	31%
NoC	6.2%	4.7%	6.2%
Memory Controller (MC)	27%	27%	27%

Table 7.1: FUB power variability.

executed and the observed PV related power variability. This dependence poses new difficulties in the WCEC estimation process since accounting for the impact of PV requires knowing the exact contribution to the power variability of each FUB. To illustrate this, we have performed an experiment using 2 synthetic software applications: a memory-intensive application and a compute-intensive one.

Intuitively, power variability caused by PV is not the same for all programs. For instance, the PV power variability in the Floating-Point Unit (FPU) has no impact on the memory-intensive benchmark: Figure 7.1 shows the PV-related power consumption variability for each FUB obtained with McPAT-PVT [177]. Probability distribution functions are normalized to make their ***y-axis*** values match the same range for visualization reasons. This allows visualizing the power variability of all processor FUB in the same plot, what would not be possible otherwise since the power Probability Distribution Function (PDF) across blocks varies significantly. We observe that for the compute-intensive benchmark the FUBs with greater contribution to the power variability are the Instruction Fetch Unit (IFU) and the FPU, whereas for the memory benchmark the FUBs with higher power variation are the IFU and Load Store Unit (LSU).

One key observation on which our proposal builds on is that the Relative Power Variability (RPV) of FUBs is constant for a given processor implementation. The relative variability of a distribution (a.k.a the coefficient of variation) is defined as the ratio of the standard deviation (σ) to the mean variation (μ). This is so because RPV depends only on the physical properties of each hardware block like its architecture and the technology library used to manufacture it. This observation allows accounting for the impact of PV in the WCEC since it enables the derivation of power quantification methods that hold regardless of the exercised workload. Two additional conclusions can be obtained from RPV values collected for the most representative FUBs of our processor design, as summarized in Table 7.1. We observe the variability of each FUB is different going from 17.5% for the RBB up to 31.2% for the L2. The second observation is that it still exists a dependence between the

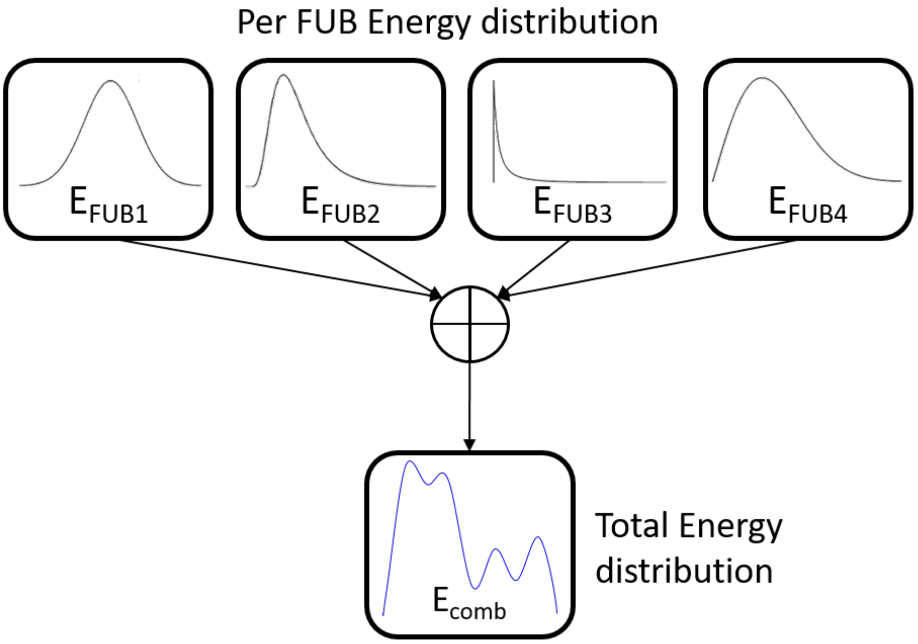


Figure 7.2: Example of different distributions per FUB.

relative power and the running workload making for instance the RPV of the ALU go from 16.2% up to 27.7% for the memory- and compute-intensive workloads, respectively. Interestingly, the dependence of the RPV with the workload disappears when considering the relative power variation per FUB access.

Also note that PV causes different impact on different FUBs, which makes the power probability distribution function vary across FUBs[1]. For instance, some FUBs can follow a Gaussian distribution while others chi-square, log-logistic, or Weibull distributions [178, 179, 180, 181]. The combination of these distributions can result in an arbitrary statistical distribution for the overall processor, see illustrative picture for 4 FUB in Figure 7.2. Building on these observations, in the next section we propose a practical methodology for estimating WCEC using energy measurements and considering the impact that PV have in the reliability of the estimates.

7.3 PV-aware Energy Modeling

Next we propose a methodology for capturing the impact of PV through measurements. We describe our methodology, its parameters, and its fitness for certification. The proposed methodology builds on current industrial practice in CPU power modeling [148], which facilitates its potential adoption.

[1] Note that our power simulator (McPAT-PVT) assumes a Gaussian distribution for all FUBs [177], hence our results and illustrative examples will reflect that, even though the actual implemented FUBs might display a different distribution.

Table 7.2: Summary of distributions used for modeling PV features.

Processor feature	**Distribution**
PV-induced power variability	Gaussian [178]
Manufacturing deviations	Gaussian fields [179]
Power and Delay due to Gate Length PV	Non-Gaussian [181]
Dynamic power per FUB	Multi-modal [180]

7.3.1 Random Nature of PV

PV impacts the physical characteristics of devices (transistors and wires), altering their nominal operation characteristics, including power and delay. PV is usually decomposed into systematic PV and random PV. The **systematic** component of PV is usually subject to strong spatial correlation across neighbor devices (transistors and wires). However, it has been shown that systematic PV impact on the different physical parameters can be accounted as an additive factor together with random PV, thus simplifying model complexity [179]. The **random** part of PV is a consequence of different uncontrolled phenomena like random dopant fluctuations. Random PV is modeled with probabilistic methods [143] that are applied either across processor units or across devices (transistors and wires). The particular implementation details of the circuits cause the impact of PV in energy distribution to vary across FUBs.

Due to the diverse nature of PV, the treatment of PV requires developing specific models to accurately capture its random impact. We list some specific methods to capture the PV impact of different parameters in Table 7.2. It follows that the actual random distribution of PV may have any shape as also illustrated in Figure 7.2. Hence, our proposal needs to build on a non-parametric statistical method. Extreme Value Theory (EVT) [55] is such a method, since it is agnostic to the particular distribution of the phenomena whose extreme behavior is to be predicted. EVT may incur some pessimism due to the fact that it fits a tail model to the maxima, as if all the population behaves as the maxima. EVT *inflates* the expected probability of maxima in its application process, thus bringing some limited, but not null, pessimism as shown in Section 7.5. Yet, EVT ends up being a reliable and tight choice as we show in this chapter.

7.3.2 The Model

Let us start representing the energy consumption of a given task as the addition of its static and dynamic components (E_{sta} and E_{dyn} as described in Section 2.4.3). Each component can be further broken down into the individual contributions across FUBs (e.g., fetch unit and L2 cache). Then, the static energy per FUB is roughly proportional to execution time and depends on the specific activity generated by each task in the case of dynamic energy. Commonly, models describe dynamic energy consumption per access type (e.g., read, write) per FUB and static energy consumption per time unit (static power) and FUB. Hence, energy consumption of a task can be described as shown in Equation 7.1, where τ_a is the task under analysis and t_a its execution time. Our processor has $\mathcal{F}$ FUBs, and each individual FUB, f, has f_y access types (e.g., reads, writes, different opcodes). Hence, P_f^{sta} stands for the static power of the FUB f and $E_{f,y}^{dyn}$ for the dynamic energy per access type y of FUB f. Finally, $Acc_a^{f,y}$ stands for the number of accesses of type y on FUB f performed by τ_a.

$$E_a = E_a^{sta} + E_a^{dyn} = \sum_{f \in \mathcal{F}} \left(P_f^{sta} \cdot t_a\right) + \sum_{f \in \mathcal{F}} \sum_{y \in f_y} \left(E_{f,y}^{dyn} \cdot Acc_a^{f,y}\right) \tag{7.1}$$

PV alters energy consumption, introducing random variations into P_f^{sta} and $E_{f,y}^{dyn}$. In particular, and based on the fact that dynamic and static energy consumption have a different nature, each component suffers a different relative dynamic and static energy variation. Still, all access types to a given component are subject to the same relative amount of variation.

Task energy accounting for PV can be derived as shown in Equation 7.2, where pv_f^{sta} and pv_f^{dyn} stand for the correction factors to account for the specific PV affecting static and dynamic energy of each FUB respectively.

$$Epv_a = \sum_{f \in \mathcal{F}} \left(P_f^{sta} \cdot pv_f^{sta} \cdot t_a\right) + \sum_{f \in \mathcal{F}} \sum_{y \in f_y} \left(E_{f,y}^{dyn} \cdot pv_f^{dyn} \cdot Acc_a^{f,y}\right) \tag{7.2}$$

The impact of PV on energy for each FUB varies due to the different devices used for their implementation. Therefore, the impact of PV on energy can be modeled by means of specific probabilistic distributions across FUBs, where each FUB is subject to a relative power variation. This variation, though different across FUBs, is regarded as homogeneous for any given FUB, so it impacts all accesses to the FUB homogeneously and does not change over time since it relates to the particular effects of PV on the chip manufactured.

Hence, pv_f^{sta} and pv_f^{dyn} can be modeled according to the underlying distribution. For instance, if such distribution is Gaussian, they would be modeled as follows:

$$pv_f^{sta} \sim \mathcal{N}\left(1, \left(\sigma_f^{sta}\right)^2\right) \tag{7.3}$$

$$pv_f^{dyn} \sim \mathcal{N}\left(1, \left(\sigma_f^{dyn}\right)^2\right) \tag{7.4}$$

where σ_f^{sta} and σ_f^{dyn} are the relative standard deviation for static and dynamic power (and energy) consumption of FUB f (e.g., 0.03 if the standard deviation for power variation is 3%).

7.3.3 Model Parameters

Table 7.3 summarizes the inputs needed in our model and how they can be derived.

Processor related parameters estimates are needed during the design and fabrication process to verify that power dissipation will not exceed the Thermal Design

Table 7.3: Parameters needed for applying the methodology.

Processor related	P_f^{sta}	Static power per FUB
	$E_{f,y}^{dyn}$	Dynamic energy per FUB per access type
	σ_f^{sta}	Standard deviation for static energy consumption per FUB
	σ_f^{dyn}	Standard deviation for dynamic energy consumption per FUB
Software related	t_a	Task's execution time
	$Acc_a^{f,y}$	Number of accesses per component and access type

Point (TDP) before manufacturing the chip. Hence, chip vendors model those parameters from information obtained in process technology tests. Once power is verified to stay below affordable levels with the electrical power model, chips are fabricated and tested. Typically, chip manufacturers use in-field data to feed models back and correct discrepancies. Hence, chip vendors can estimate with high precision the power parameters needed in Equations 7.2, 7.3, and 7.4.

Software related parameters can be measured during software tests by means of the Performance Monitoring Unit (PMU). Current PMUs offer several Performance Monitoring Counters (PMCs) to monitor a large variety of events, including a breakdown for each type of operation. PMCs allow monitoring events with high accuracy, yet some residual error may exist due to, for instance, the fact that events are counted with some little slack between the time they occur and the time they are effectively counted in the corresponding PMC. However, this effect may distort the statistics for, at most, few tens of cycles, thus few tens of nanoseconds. Given that typical execution times for tasks in autonomous systems are in the order of milliseconds, the inaccuracy introduced by PMUs in terms of both timing and energy is fairly below 0.01%, which is completely negligible in comparison with the precision of the power delivered by the power supply, or the energy consumed by mechanical components in the system.

7.4 WCEC Estimation Methodology

WCEC estimation is useful for application developers or system integrators that need to provide guarantees about their software being compliant with strict energy consumption constraints for autonomous systems. Our WCEC estimation approach, whose overall process we summarize in Figure 7.3, consists of two main steps: (1) collecting representative energy measurements of the task and (2) estimating the energy budget needed so that it cannot be exceeded with a relevant probability.

Measurement collection (sampling). Once the task has been executed and software-related parameters obtained through the PMU, our method produces energy measurements accounting for the impact of PV. To that end, we perform a Monte-Carlo experiment where pv_f^{sta} and pv_f^{dyn} in Equation 7.2 are sampled from their reference distributions. Each observation of the Monte-Carlo experiment (i.e., $o \in \mathcal{O}$) delivers specific $pv_{f,o}^{sta}$ and $pv_{f,o}^{dyn}$ values for each FUB $f \in \mathcal{F}$. These are used to

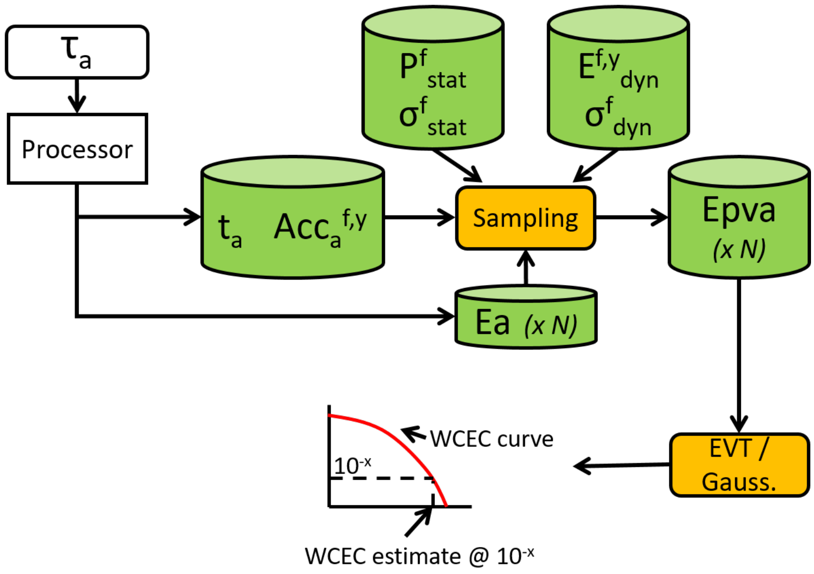


Figure 7.3: WCEC estimation process and inputs it builds on.

produce a specific PV corrected energy sample (Epv_a^o) from the energy sample (E_a) of the task under analysis τ_a.

WCEC distribution. In order to derive the WCEC estimates, a method suitable for estimating extreme behavior based on the observations of the central behavior is needed. We regard EVT [55] as a convenient method for that purpose. EVT has already been used successfully in the context of Worst-Case Execution Time (WCET) estimation, resulting in probabilistic WCET estimates [33, 51, 50] and we refer the reader to Section 2.2 for details on EVT.

In particular, in applying EVT to WCEC estimation we resort to the EVT application process in [33], which carries the following application requirements: it applies to independent data and processes and when an exponential tail is guaranteed to be a reliable upper-bound. Energy measurements in the sample correspond to independent and identically distributed observations of the same phenomenon (random variable) by construction of the process studied (energy consumption variation due to PV) and measurement protocol used (not carrying any state across measurements). From this observation, it follows that no dependence exists across input measurements, which we empirically assess with proper Independence and Identical Distribution (I.I.D.). tests [58, 57], which are a prerequisite for the reliable application of EVT.

The minimum sample size for a reliable application of EVT is only dictated by EVT itself. We start generating 1,000 energy measurements as initial sample size and increase the sample size whenever the method requests it. In this work in particular some of the experiments required 2,000 measurements, hence we used 2,000 measurements for the sake of homogeneity.

Accounting for multiple program inputs. Our methodology covers a specific set of input values for the program. However, test campaigns need to account for different operation conditions, which are modelled using multiple input sets for the program under analysis. The way to proceed resembles the approach followed for WCET (timing) estimation [56]. Hence, the methodology above needs to be applied independently for each set of input values, and EVT used in each individual set of measurements for a given input set. Then, the different WCEC distributions obtained need to be combined using the *max envelope* operator which, for each exceedance probability selects the highest energy value across all WCEC distributions, thus delivering the tightest WCEC distribution that upper-bounds all those for each individual input set.

7.4.1 WCEC Interpretation and Safety Standards

Once we obtain the WCEC distribution, we can select as WCEC estimate the value whose exceedance probability is sufficiently low. Since the only source of variation is PV and it changes across chip units, a given exceedance probability relates to the probability of having a processor unit that may exceed such energy value systematically due to its specific PV.

This approach fits current V&V practice according to safety standards (e.g., IEC-62304 for medical devices and IEC-61508 for industry). Safety standards require the quantification, or the qualitative assessment, that the risk of hazardous situations is below tolerable rates. In general, safety goals and safety requirements are defined with the aim of mitigating – rather than eliminating – the risk that hardware or software misbehavior causes a system failure. As an illustrative example, for automotive, ISO-26262 stipulates the maximum allowable likelihood of occurrence of random hardware faults. In doing so, ISO-26262 acknowledges that safety techniques cannot achieve full coverage, allowing different diagnostic coverage. For instance, for the highest-criticality items (ASIL-D), ISO-26262 requires proving residual failure rates below 10^{-7} for diagnostic coverage above 99.9%.

Overall, the interpretation of the energy exceedance probability matches that of defective hardware components (e.g., the probability of having a defective processor or a defective wheel). For instance, we can set the exceedance probability down to 10^{-9}, thus meaning that at most 1 every 10^{9} processors may lead to exceeding the WCEC estimate for this task.

7.5 Experimental Results

7.5.1 Evaluation Framework

Architectural, power, and PV models. While processor vendors have the data needed by our model, this information is usually not released for commercial processors for autonomous systems. Hence, we build on SoCLib [99], a cycle accurate simulator, to model the timing behavior of a LEON4 processor, whose block diagram we replicate from Chapter 3 in Figure 7.4 for ease of reading.

We integrated McPAT-PVT [177] power estimation methodology into SoCLib to collect energy and power measurements. McPAT-PVT is an extended version of the

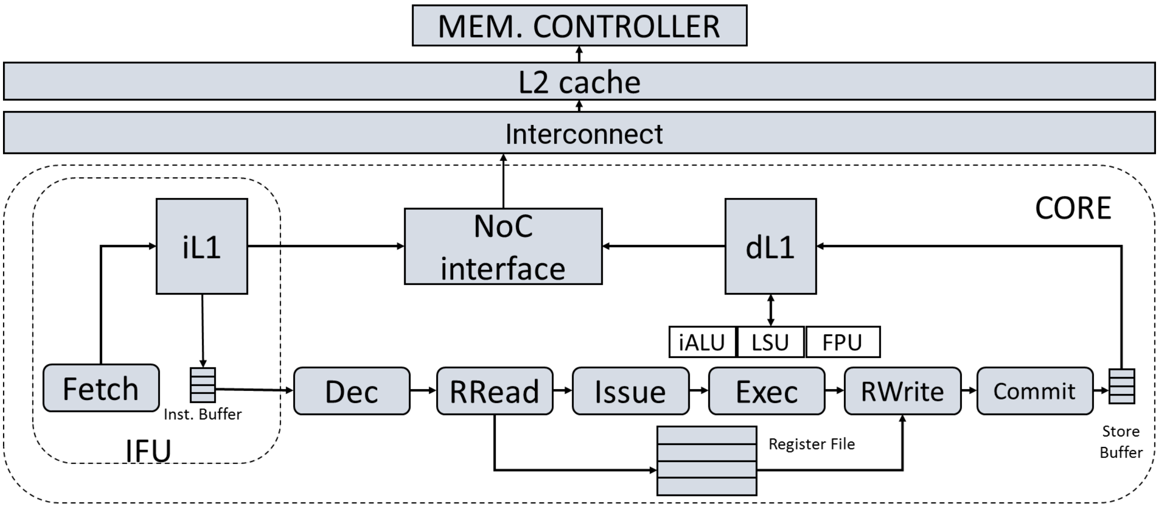


Figure 7.4: Block diagram of our reference architecture.

McPAT tool [101] that allows accounting for the impact of PV in power measurements. For our experiments, we model a process technology of 22 nm, an operating voltage of 0.9 V, and an operating frequency of 700 MHz. Note that, the methodology is architecture and benchmark agnostic and our set-up just a representative example of the real-time domain.

Benchmarks and case studies. **1)** We evaluate two space case studies: DEBIE and OBDP. **2)** We also use EEMBC automotive benchmarks [106] as reference benchmark suite, since they represent a number of critical real-time functions of some automotive systems. In particular we use cacheb, a2time, aifftr, aifirf, aiifft, basefp, bitmnp, canrdr, idctrn, iirflt, matrix, pntrch, puwmod, rspeed, tblook, ttsprk. We refer the reader to Chapter 3 for further details on the experimental setup.

Statistical Characterization of PV. We randomly generate a population of processor instances, N_p, whose FUB's PV behaves according to the specific distributions that would be provided by the processor manufacturer for real processors. In our setup used for illustration and evaluation purposes, we obtained those values from the McPAT-PVT power estimation tool due to the lack of this information from a real processor. However, in a practical case, such information would be provided by the chip vendor. Note, however, that our methodology holds regardless of the actual values used and hence, the representativeness of McPAT-PVT values, although it has already been discussed in [177], has no impact on the method proposed in this chapter. This approach delivers N_p independent energy measurements per benchmark that resemble the chip-to-chip energy variations. Unless stated otherwise, we focus on Gaussian distributions in the remaining of the chapter.

Table 7.4: Maximum observed energy, and pWCEC (in μJ) with PV.

bench	**MAX**	ΔE_{PV}	**pWCEC** EVT(10^{-7})	Δ^{EVT}_{pWCEC}	**pWCEC** Gauss	Δ^{Gauss}_{pWCEC}
cacheb	202.7	92.9 %	239.4	18.1 %	249.2	23.0 %
matrix	8259.2	92.7 %	9716.5	17.6 %	10031.2	21.5 %
aifftr	2070.1	95.0 %	2520.3	21.7 %	2486.2	20.1 %
pntrch	51.4	98.2 %	58.2	13.2 %	62.2	21.1 %
rspeed	15.0	104.7 %	18.4	22.8 %	17.9	19.8 %
puwmod	54.4	101.2 %	67.0	23.1 %	65.4	20.2 %
aifirf	39.4	94.3 %	53.3	35.4 %	48.7	23.8 %
aiifft	1914.5	94.7 %	2051.9	7.2 %	2301.6	20.2 %
a2time	25.6	98.8 %	33.1	29.3 %	31.5	23.1 %
idctrn	348.2	92.2 %	437.9	25.7 %	429.0	23.2 %
iirflt	38.3	103.5 %	49.5	29.2 %	46.5	21.5 %
basefp	59.7	106.5 %	64.5	8.0 %	70.7	18.4 %
bitmnp	196.8	98.2 %	231.1	17.4 %	236.1	20.0 %
tblook	17.1	100.9 %	21.6	26.3 %	20.8	21.5 %
canrdr	36.6	99.8 %	46.8	27.8 %	44.3	21.1 %
ttsprk	37.6	103.7 %	46.2	22.9 %	44.4	18.4 %
OBDP	143817.0	94.6 %	205486.1	42.9 %	176506.2	22.7 %
DEBIE	228420.7	100.2 %	263980.0	15.6 %	269121.3	17.8 %

7.5.2 PV-generated power variability

In our setup, from the execution of each benchmark in the simulator we obtain the number of accesses to each FUB ($Acc_a^{f,u}$) and the task's execution time (t_a), which we fed into the power model of McPAT-PVT. McPAT-PVT provides static power per FUB and cycle (P_f^{sta}) and dynamic energy per access type per component ($E_{f,y}^{dyn}$). Building on these parameters, we obtain the power dissipation per component as well as a power variation σ per component due to PV, as presented in Section 7.3.

The first two columns (after the benchmark names) in Table 7.4 show the absolute maximum energy consumption per benchmark, and the magnitude of the impact of variations, labeled as ΔE_{PV}. The latter is computed as $\frac{max-avg}{avg}$. We observe increments as high as 117% (the maximum is ≈2.2x the average), while average variations are of 100% (≈2x). This means that the maximum value observed is, on average, 2x times the average, thus further emphasizing the importance of accounting for PV in WCEC estimation.

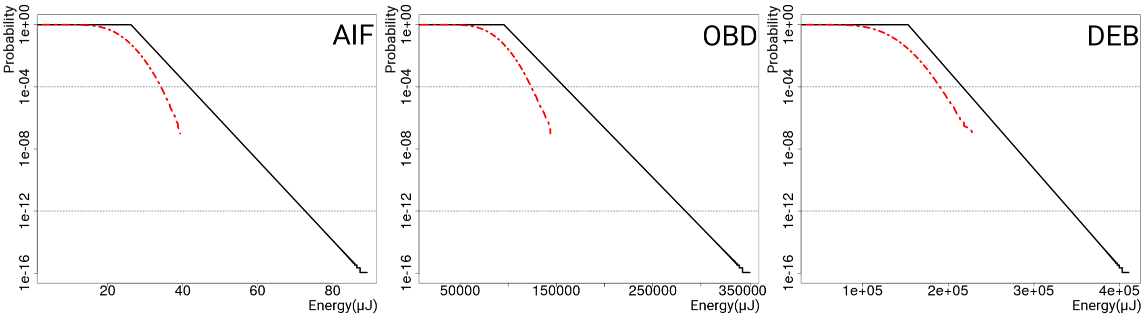


Figure 7.5: pWCEC distribution in μJoules and empirical CCDF of the PV-adjusted energy measurement.

7.5.3 Probabilistic WCEC Estimates

As presented before, starting from a set of measurements $o \in O$ of the energy for the modeled processor unit under analysis , E^o_a, we use specific statistical correction factors $pv^{sta}_{f,o}$ and $pv^{dyn}_{f,o}$ values for each FUB f to produce a PV corrected energy sample (Epv^o_a). This sample is passed as an input to EVT to generate a Probabilistic Worst-Case Energy Consumption (pWCEC) estimation that describes the probability of an arbitrary processor unit to exceed an energy consumption value.

We carried out this process for all the reference benchmarks used in this chapter, and we show results for a representative subset of them. In particular, Figure 7.5 shows 3 plots – 2 European Space Agency (ESA) applications, and the EEMBC with the highest pWCEC over-estimation (aifirf) – with their corresponding pWCEC distributions. Red dashed lines correspond to the empirical Empirical Complementary Cumulative Distribution Function (ECCDF) of the measurements, whereas straight black lines stand for the pWCEC distributions.

To provide evidence on the confidence in deriving WCEC estimates, we collected 10^7 measurements for each benchmark. Note that performing such an experiment is not needed (and it is infeasible in the general case). We use it for comparison purposes and hence, pWCEC is estimated with 2,000 measurements. For the lowest probability for which we measured the actual distribution, 10^{-7}, pWCEC curves are 22.6% higher than observations on average.

We observe that pWCEC distributions upper-bound observed energy consumption for all benchmarks, and gently follow the observed distributions. We also observe that the slope (the vertical variation) of the observed distribution is also gentle. This shows that the impact of PV is high and emphasizes the importance of properly accounting for PV in the process of WCEC estimation, as PV can produce large energy variations.

7.5.4 Multitask Workloads

We execute a set of four tasks under analysis in a multi-core environment, and apply the previously described methodology to such setup. In particular, we created 4 workloads that cover all EEMBC benchmarks. These workloads are W_1 ={*a2time, idctrn, aifftr, aifirf*}, W_2 ={*aiifft, basefp, bitmnp, cacheb*}, W_3 ={*canrdr, iirflt, matrix,*

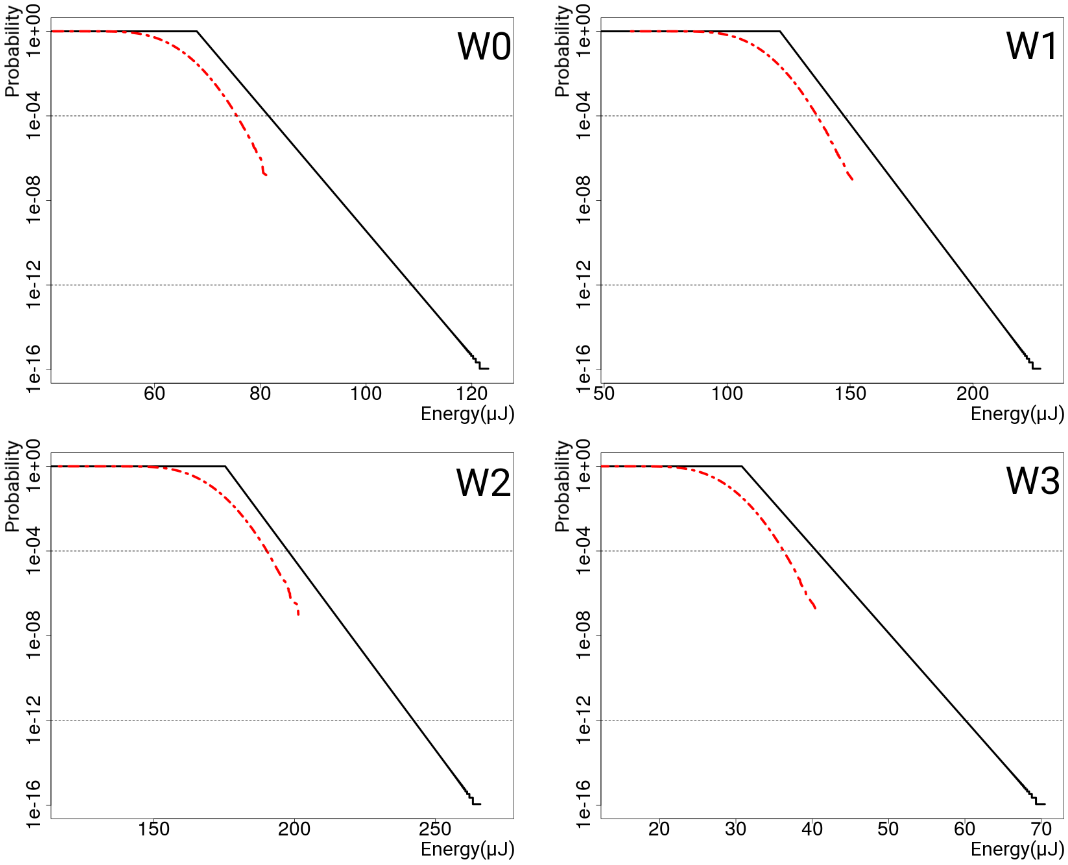


Figure 7.6: pWCEC estimates for multi-core workloads.

pntrch}, and W_4 ={*puwmod, rspeed, tblook, ttsprk*}.

Figure 7.6 shows pWCEC results like those in Figure 7.5 but where each plot represents the simultaneous execution of 4 EEMBC benchmarks. We see that EVT successfully upper-bounds the maximum energy consumption observed, with an overestimation that ranges between 21.4% (W_3) and 31.8% (W_1) with an average of 26.04% with respect to maximum observed value. Although the overestimation is not high, there is margin for tighter bounds which could be obtained by tuning and changing the EVT algorithm, but such objective remains out of the scope of this work.

7.5.5 Comparing EVT vs Gaussian approach

So far we focused on the All-Gaussian PV setup, in which all sources of PV follow a Gaussian distribution. In this scenario, intuitively, Gaussian modeling instead of EVT seems a better fit. We have compared the pWCEC estimates produced with both. The last two columns in Table 7.4 shows the absolute estimate produced by Gaussian and its degree of overestimation with respect to the maximum observed energy value. For this experiment, in the $\mu + N \cdot \sigma$ formulation, we set $N = 5.33$ so that the exceedance probability is below the chosen threshold. As it can be observed, Gaussian modeling produces quite similar results to those of EVT (22.6% vs 21.2%

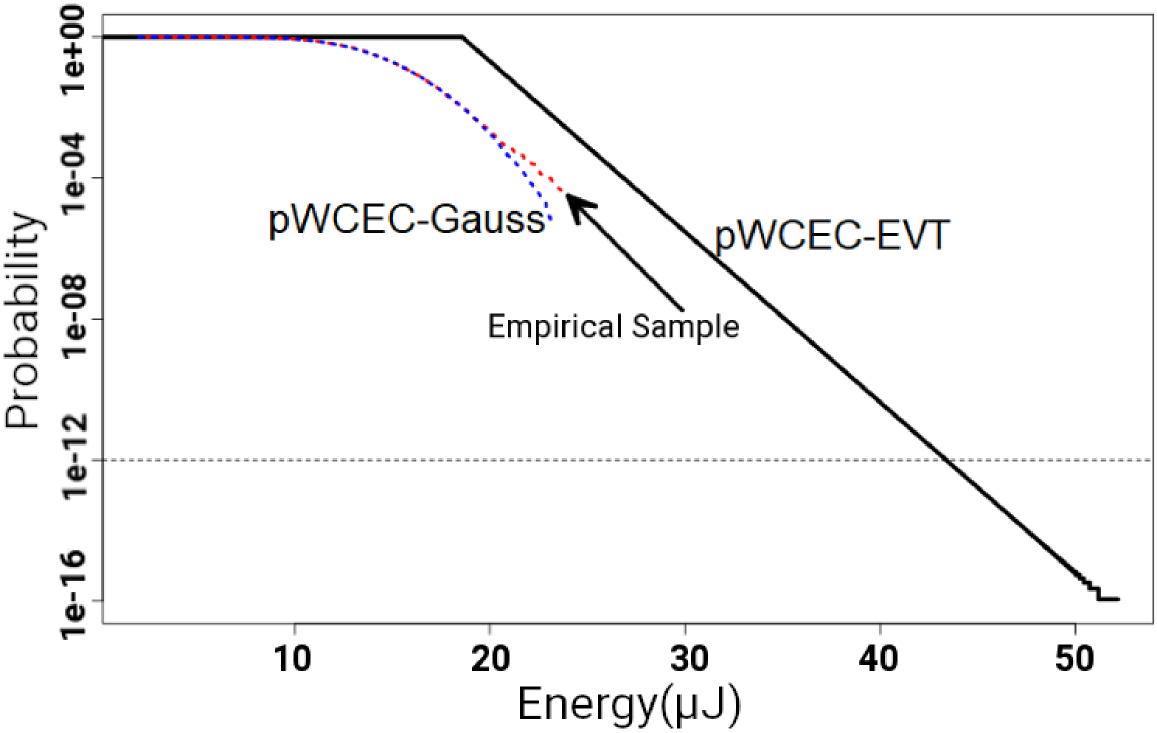


Figure 7.7: pWCEC curves and ECCDF of the samples.

larger than maximum observed values on average).

However, Gaussian modeling provides poor (unreliable) estimates as soon as the underlying PV does not follow a Gaussian distribution, as it has been shown to be the case for some circuit parameters (see Section 7.2). To provide evidence on this matter, we repeated the experiments but focusing on the mixed-distribution PV setup introduced in Section 7.5.1. In particular, for this experiment we focus on `a2time` benchmark.

The resulting pWCEC estimates can be seen in Figure 7.7. In blue we have the pWCEC-Gaussian distribution, in red the empirical data observations, and in black the pWCEC-EVT distribution. We can see that, as the exceedance probability decreases, the Gaussian model losses fitness and starts under-estimating the WCEC. In contrast, the EVT curve properly upper-bounds the sampled data.

7.6 Summary

We have analyzed PV impact on the processor energy consumption, and presented a methodology based on statistical-modeling that deals with PV during the WCEC estimation process of autonomous systems. This enables the estimation of WCEC by accounting for the probabilistic nature of PV and using probabilistic approaches for WCEC estimation, such as EVT. Our results show that the impact on energy of PV is large, and can be appropriately bounded with probabilistic means.

Chapter 8

Detecting and Hampering Worst-Case Power Peak Events during Testing

> “*Alea iacta est.*”
>
> — Gaius Julius Caesar

8.1 Introduction

The verification and validation process of embedded critical systems requires providing evidence of their functional correctness, and also that their non-functional behavior stays within limits. In this chapter, we focus on power peaks, which may cause voltage droops and thus, challenge performance to preserve correct operation upon droops. The use of complex software and hardware in critical embedded systems jeopardizes the confidence that can be placed on the tests carried out during the campaigns performed at analysis. This occurs since it is unknown whether tests have triggered the highest power peaks that can occur during operation and whether any such peak can occur systematically. In this chapter we propose the use of time randomization, already used for timing analysis of real-time systems, as an enabler to guarantee that (1) tests expose those peaks that can arise during operation and (2) peaks cannot occur systematically inadvertently.

In embedded critical systems, the Validation and Verification (V&V) process builds not only on collecting evidence about their correct functional behavior, but also about their non-functional behavior including timing, power and temperature among other concerns. Due to economical and practical reasons, industry often relies on measurement-based approaches to derive such evidence [182].

The increasing performance needs in embedded critical systems are satisfied at a reasonable cost by using advanced (complex) hardware platforms. In those platforms, deriving test cases that trigger worst-case conditions becomes increasingly difficult for end users. For power verification, defining appropriate test cases and input vectors is critically important to identify whether (high) power peaks can occur and whether they can occur systematically [156]. Power peaks may lead to sporadic or frequent voltage droops that need lowering speed or stalling execution to preserve correctness [183, 184, 185], hence impacting timing of tasks in general, and real-time tasks in particular. For instance, power peaks may depend on the simultaneous occurrence of a number of events in cores, caches and on-chip interconnects, whose fine-grain control cannot be practically exercised. Thus, by analyzing power traces, end users are generally unable to tell whether higher power peaks can occur and, if so, whether they could occur systematically. The feasibility of triggering worst-case power scenarios determines whether real-time tasks, and especially those with some form of criticality (e.g., due to safety or security), can be successfully verified or not.

Recently, injecting randomization at hardware and software level has been proposed as a means to facilitate timing analysis of critical real-time tasks [186, 50] by (1) breaking systematic pathological timing behaviors, so that increasingly higher execution times have rapidly decreasing probabilities, and (2) making bad (long) execution times not to occur during test campaigns with probabilistically low bounds. The latter simplifies deriving the probability of occurrence of high execution times (i.e., those beyond the maximum observed execution time) with statistical means such as Extreme Value Theory (EVT) [55].

However, to our knowledge the applicability and the specific application process of time-randomization solutions to mitigate power peaks and reduce the cost of power testing campaigns have not been explored. To cover this gap, we explore whether the randomization injected in Time-Randomized Processors (TRP) [186] can be used in embedded systems to expose pathological worst-case power profiles and break their systematic occurrence, so that their impact is limited and can be properly accounted for.

8.2 Challenges of Power Verification in Complex Processors

8.2.1 Power Delivery Network Sizing

Power Delivery Networks (PDNs) in processors are typically designed to serve enough power ”in most cases”, but due to efficiency reasons, they are not designed to meet the power requirements in the absolute worst case, since it may occur only occasionally. For instance, Figure 8.1 shows the power profile of an arbitrary benchmark running in a simple and a more complex processors. We observe that power variation is significant and the relative difference between the absolute worst case observed and the typical case is large, and it increases in absolute terms for increasingly complex designs. Thus, sizing PDNs for the absolute worst case would result in a waste of resources.

Overall, instantaneous power demand may surpass the capacity of the PDN, thus leading to a scenario where circuits become under-powered during relatively short time intervals, until the power demand decreases. In such scenario, voltage decreases to

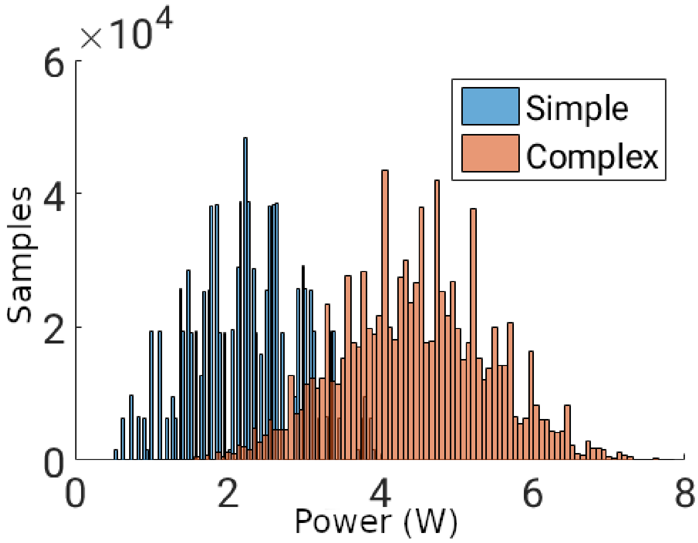


Figure 8.1: Histogram of instantaneous power measurements in a simple and a complex processor. Synthetic experiment showing the increase in power variability as processor complexity rises.

levels where correct operation cannot be preserved – often referred to as voltage droops – and actions such as decreasing operating frequency must be taken to decrease power demand and preserve correct operation [183, 184, 185]. While the effect of droops is relatively small in high-performance systems, in critical systems their impact on metrics like worst-case timing and power budgeting can be high.

8.2.2 Critical Real-Time Systems Verification

Processor verification is typically performed using power viruses [187] to characterize the corner power cases of the processor, size its PDN and accommodate mechanisms able to detect overly high power consumption and throttle (or even stop) operation to preserve processor physical integrity. For embedded critical (real-time) systems verification, processor integrity is not a concern, since appropriate means have already been set by the chip manufacturer. However, voltage droops as well as overly high sustained power dissipation may lead to performance issues due to, for instance, performance throttling. Authors in [184] show that a usual solution would be decreasing operating frequency down to its minimum (e.g., 1/32 of its maximum value) and increase it progressively as long as power demand does not exceed affordable limits. Hence, assessing during system analysis phases whether power peaks can occur, their magnitude and their frequency is critically important to evaluate whether timing bounds will be respected. However, end users often lack the knowledge of how power peaks arise in a specific processor, and lack the means to assess whether applications can trigger them. This may jeopardize the complete timing verification of the system if the impact of voltage droops is not properly accounted for during testing.

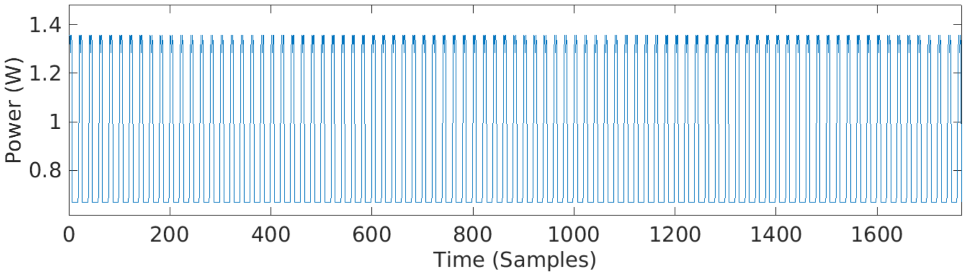


Figure 8.2: Power profile on a conventional (simple) architecture when running two unsynchronized benchmarks.

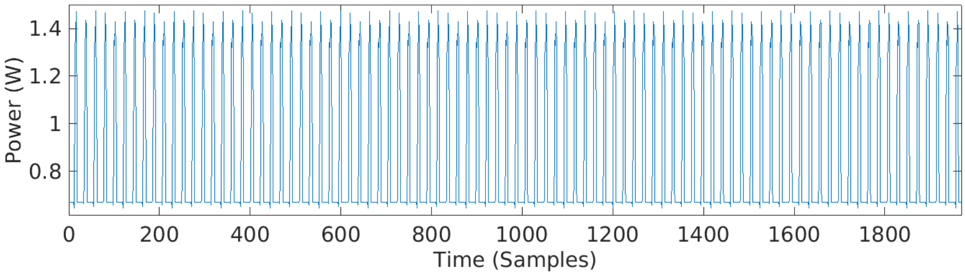


Figure 8.3: Power profile on a conventional (simple) architecture when running two synchronized benchmarks.

8.2.3 An Illustrative Example

Let us consider a simple example with two programs running simultaneously in different cores of a multi-core processor. Figure 8.2 shows their joint power profile, with power measured every 43 ns (see Section 8.4) and the x-axis showing each of these observations over time. The two programs iteratively spend some time performing local (in core) computations, followed by a period of sustained memory write operations. As shown, frequent power peaks due to memory accesses interleave accesses of both programs and stay below 1.4 W.

In a second experiment, we modified one of the benchmarks introducing few delays in between their memory accesses, thus effectively decreasing its average power dissipation and without impacting its individual maximum power dissipation. As shown in Figure 8.3, the time alignment of the power peaks changes slightly and, despite the overall average power dissipation decreases, the power peaks increase in magnitude, being above 1.4 W sustainedly. If the PDN of this processor could only afford up to 1.4 W of power, we would move from a scenario with no voltage droops to a scenario with systematic droops. And potentially, the latter scenario could not occur during testing, which would lead to the risk of missing deadlines systematically due to frequent unforeseen voltage droops.

In this particular example, we first created the two programs and run them without any specific synchronization. Then, since the platform used is a performance simulator, we had access to the internals of the architecture and could debug why some events were not occurring simultaneously and applied reverse engineering to

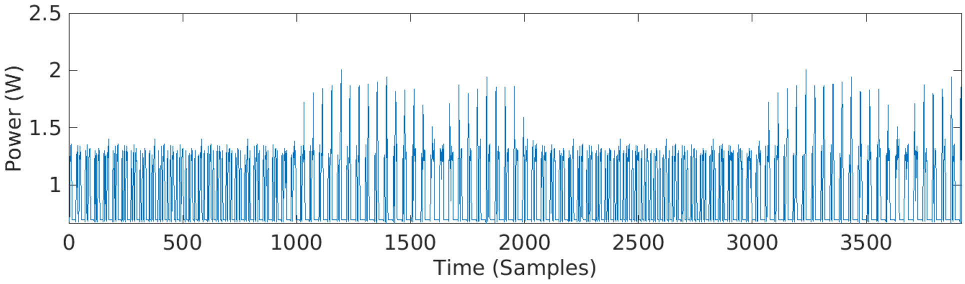


Figure 8.4: Power profile on a conventional (complex) architecture when running four benchmarks.

cause a pathological systematic behavior where events align perfectly and lead to higher power peaks. However, in the general case this is not doable. In fact, we repeated the same experiment modeling a more complex processor with 4 cores instead of 2, allowing multiple memory requests in flight and increasing store buffers and, despite having full access to the architecture in the simulator, we were unable to exercise the control needed to synchronize events. Figure 8.4 shows the power profile of the execution of four benchmarks in the 4 cores and, as shown, some peaks occur from time to time, but it is unclear whether higher peaks can occur and whether they can occur systematically.

In summary, in complex hardware with time-deterministic behavior it cannot be assessed whether tests trigger the highest power peaks and whether those can occur systematically. This jeopardizes the confidence that can be obtained from test campaigns with uncertainty on the risk of deadline violations due to voltage droops since they cannot be bound reliably.

8.3 Time-Randomization for Power Analysis

Power variation is highly correlated with the same events that create timing variation, which include cache hits/misses, arbitration delays in shared resources, variable delays in queues, etc. Time-randomization, either implemented by hardware or software means [186], allows exploring, for timing analysis purposes, the different outcomes of those events enforcing probability distributions that hold during analysis and operation. In this section, we analyze how time-randomization serves also the purpose of exploring power peaks, either in frequency or in magnitude, as well as the limits of time-randomization.

8.3.1 Event Alignment

Power peaks emanate from the simultaneous occurrence of multiple high-power events. Next we review how events relate to each other and the influence that time-randomization may have on them:

Potentially aligned events. Some events may align under certain conditions, such as those shown in Figures 8.2 and 8.3. By introducing time-randomization at a

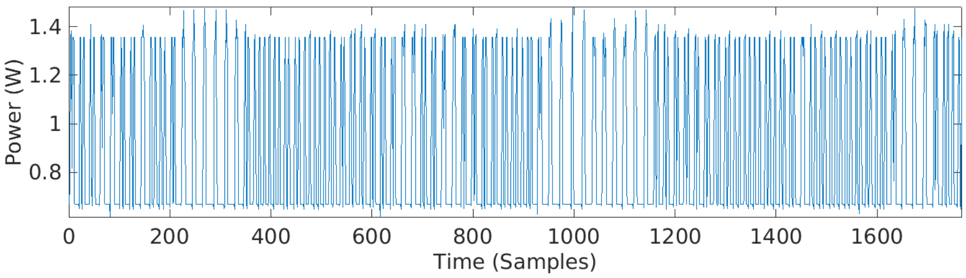


Figure 8.5: Power profile on a time-randomized (simple) architecture when running two unsynchronized benchmarks.

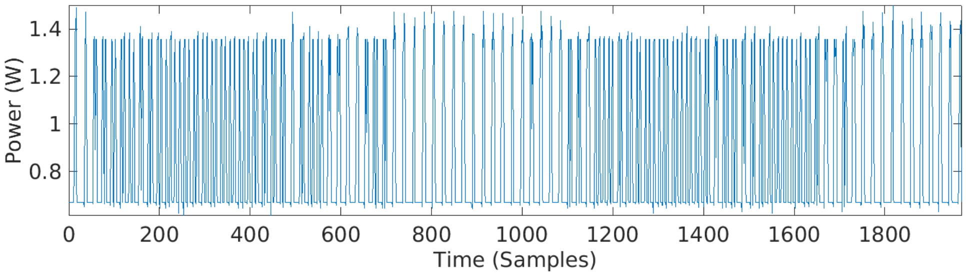


Figure 8.6: Power profile on a time-randomized (simple) architecture when running two synchronized benchmarks.

fine granularity (by making arbitration delays vary by few cycles, and making some cache hits become misses and vice versa) the power-hungry events that might concur are enforced to concur with some probability. This contrasts with the scenario drawn for time-deterministic platforms, in which events may never (or frequently) align with specific tests, and whose behavior can change completely during operation simply because the initial state of the processor or memory varies subtly. Overall, time-randomization allows making a probabilistic argument on the appearance of such type of events, and more importantly, make them not occur systematically.

Figures 8.5 and 8.6 show the same experiments done for Figures 8.2 and 8.3, but carried out on a time-randomized platform. In particular, random placement and replacement caches as well as random bus and memory controller arbitration are implemented, as detailed later in Section 8.4 [186]. As shown, both power profiles show those peaks occurring when power-hungry events align, but they do not occur systematically. Moreover, power profiles are probabilistically almost identical among them since event alignment occurs with similar probabilities across experiments. Therefore, power peaks are naturally exposed and can be accounted for conveniently.

Never aligned (or nonexistent) events. Some events may never align in time-deterministic systems because, for instance, the initial conditions that trigger their alignment never occur during operation. In this case, the difficulties emanate from the fact that, upon not observing their alignment, end users lack information on whether they can never align, whether tests simply failed to align them (as in Figure 8.2), or even whether higher peaks exist. Time randomization, instead, will

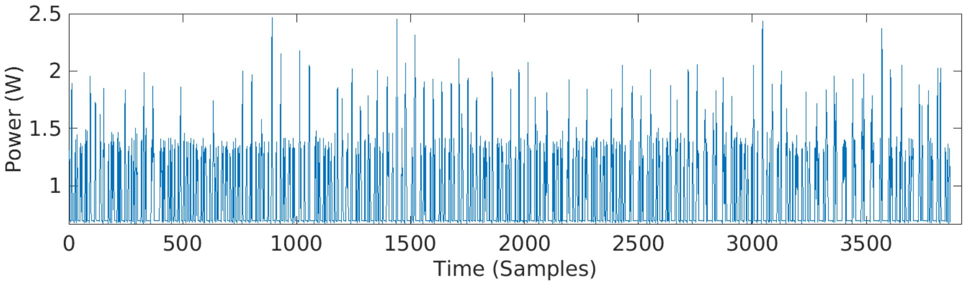


Figure 8.7: Power profile on a time-randomized (complex) architecture when running four benchmarks.

make those events align with a probability, so they are observed and accounted for (perhaps pessimistically). Even when time randomization does not make them align, then confidence is gained that they cannot align with high probability.

It may also occur that time randomization causes some high-power events that would not occur without time randomization (e.g., causing few additional cache misses). While this effect is known to be very low since time-randomization degrades performance negligibly in the average case [186], it may lead to some pessimism due to triggering peaks that would not exist ever without time randomization.

Systematically aligned events. Some events may be highly aligned leading to systematic power peaks. If randomization may unalign them, it will allow reducing their impact due to voltage droops. Instead, if they cannot be unaligned because their occurrence is caused by events with no practical variability (e.g., sustained floating point operations), then randomization brings no quantitative difference. Yet, randomization brings confidence on the fact that high power peaks are observed during testing, so that their worst impact can be reliably predicted.

Overall, while time-randomization will have little influence in the average power dissipation and average number of power peaks across programs, it has two key advantages:

1. It guarantees probabilistically that peaks that can occur during operation are observed during testing.
2. If systematic behavior can be broken, it is effectively broken, thus allowing to account for peaks probabilistically without having to resort to overly pessimistic assumptions.

8.3.2 An Illustrative Example

For the sake of completeness, we have also repeated the experiment on the complex processor with time-randomization. As shown in Figure 8.7, power peaks are naturally exposed. In fact, some peaks are clearly higher than those observed in the time-deterministic setup. Thus, time-randomization allows accounting for their occurrence. Instead, in the case of time-deterministic platforms, it is unknown whether they can occur in practice and, if so, whether they can do it systematically, thus defeating any confidence had on the test campaign.

8.3.3 On Predicting Power Peaks

With time-randomized platforms we can use power measurements to predict both (1) peaks magnitude and (2) frequency. For that purpose, we build on the Measurement-Based Probabilistic Timing Analysis using Coefficient of Variation (MBPTA-CV) method, given that the properties needed for its input data are preserved:

- Independence and Identical Distribution (I.I.D.): MBPTA-CV inherits from the use of EVT the need for I.I.D. input data. While power measurements are not fully independent in practice at any time granularity, they quickly become independent since processor events last typically up to some tens of nanoseconds, which is the same order of magnitude of peaks duration to cause voltage droops. Hence, measurements at short distance are already independent. Moreover, EVT, in practice, does not need I.I.D. measurements but I.I.D. maxima which means that dependencies across those values not belonging to the upper-tail of the distribution are irrelevant [51]. In any case, input samples passed to MBPTA-CV need to be tested against I.I.D. statistical properties for a reliable use of MBPTA-CV.
- Exponentiality: MBPTA-CV fits exponential tails, thus discarding heavy tails. This is only a reliable choice for distributions that have a maximum value, despite such maximum can be unknown. In the case of power, either due to temperature limitations or due to power supply limitations, a maximum power is known to exist and hence, the premise for the use of MBPTA-CV holds.

We identify two different ways of using MBPTA-CV in the context of power verification:

- High power peaks determination to retrieve either the highest power value that could occur with a meaningful probability (e.g., that could only be exceeded with a probability below 10^{-12} per time unit). Also whether a particular power value could be exceeded with a probability higher than a given threshold (e.g., whether a peak causing a voltage droop occurs with a probability above 10^{-12} per time unit). Note that the time unit relates to the granularity at which voltage droops may occur (e.g., a peak of few picoseconds would be irrelevant).
- Estimating the number of times that a given threshold is exceeded. By measuring the number of times the threshold is exceeded in each run of the program or the workload, we can estimate the highest number of peaks we can expect whose exceedance probability is below a given threshold (e.g., how many power peaks we can expect so that a higher number of peaks is expected less than once every 10^{12} runs). In this case, by using an upper-bound of the time to recover from a voltage droop (e.g., 100 ns), we can increase the Worst-Case Execution Time (WCET) estimate accounting for the maximum number of peaks expected (e.g., 50 peaks) multiplied by the recovery time for any such peak.

8.4 Quantitative Assessment

In this section we show a practical application of time-randomized platforms together with MBPTA-CV for power verification.

8.4.1 Experimental Setup

For the details on our experimental setup in this section we refer the reader to Chapter 3. As processor model we use the Cobham Gaisler NGMP [94]. For the examples in Sections 8.2 and 8.3, we use a simple version with only 2 cores, 2-entry store buffers and up to one outstanding core request (L1 cache miss). For the complex version, we use the full 4-core setup with 8-entry store buffers and up to 6 outstanding core requests (typically non-blocking store operations).

The time-randomized setup uses random modulo placement L1 caches, random hash placement L2 cache, random replacement in all caches, and random permutation arbitration in the bus and memory controller [37]. The time-deterministic setup, instead, uses modulo placement and Least Recently Used (LRU) replacement caches, and round-robin bus and memory controller arbitration.

Apart from the benchmarks used for the previous examples, we use the EEMBC Automotive reference benchmark suite [106], which includes a number of representative applications for critical real-time systems.

8.4.2 Power Verification

First, we evaluate the highest power peak expected. Whether this analysis needs to be done at chip level (so for the full workload) or at core level (so for each benchmark individually) relates to the organization of the PDN, and so the region where voltage droops can occur. For instance, in the case of a multi-core with an independent PDN for each core, it might be more appropriate for the methodology to require individual per core analysis of peaks, while with a shared workload or PDN, whole-system peak analysis might be more suitable. However, this is irrelevant for the application of the methodology. For instance, Figure 8.8 shows the power profile of the whole chip for one run of a 4-benchmark workload on the time-randomized complex setup. As shown, the randomized behavior of the power peaks can be noticed regardless of the integration level.

For simplicity and illustration purposes, the rest of the discussion is done for individual benchmarks executed in a single-core. Figure 8.9 shows the probabilistic power distribution for `aifirf` benchmark in μW, in the form of the Complementary Cumulative Distribution Function (CCDF). The red dashed line corresponds to the actual measurements, the black thick line to the estimated high power distribution, and the blue thin lines to the 95% confidence interval. As shown, by having the full distribution, we can obtain the power value for any exceedance probability or the exceedance probability for any power value.

While estimating the highest power peak for a given program may have several applications, in the context of critical real-time system we regard as more relevant

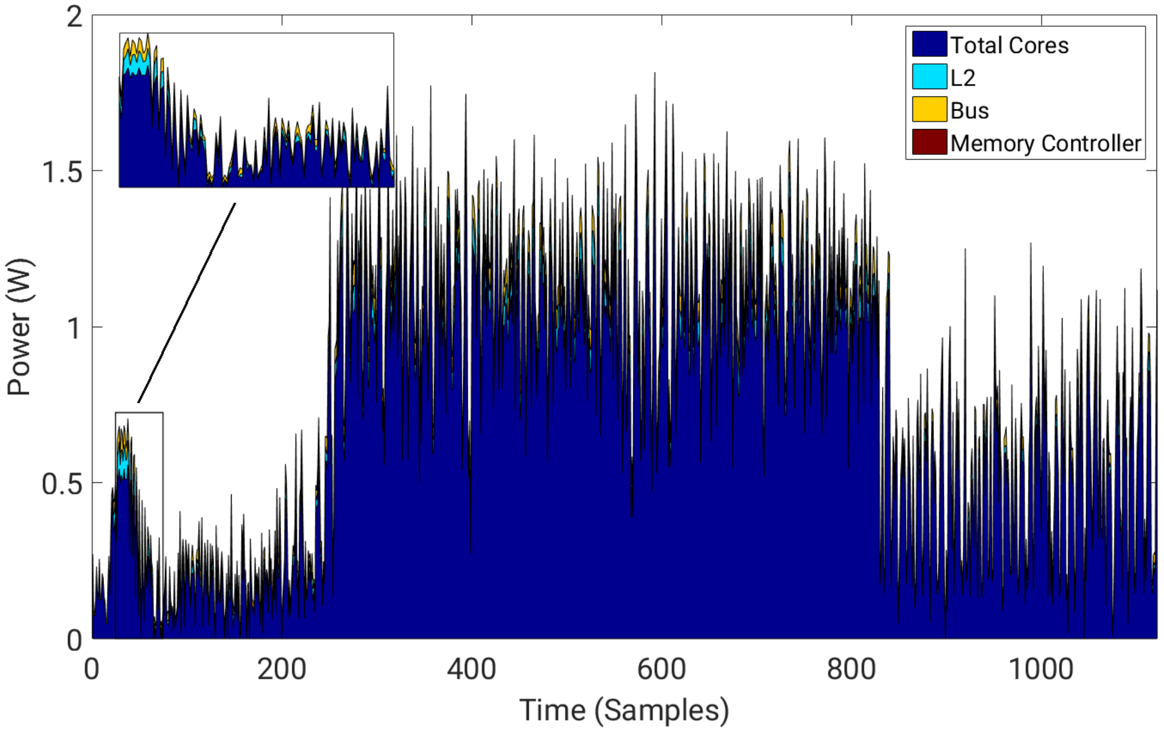


Figure 8.8: Power dissipation over time of 4 different EEMBC in a randomized hardware.

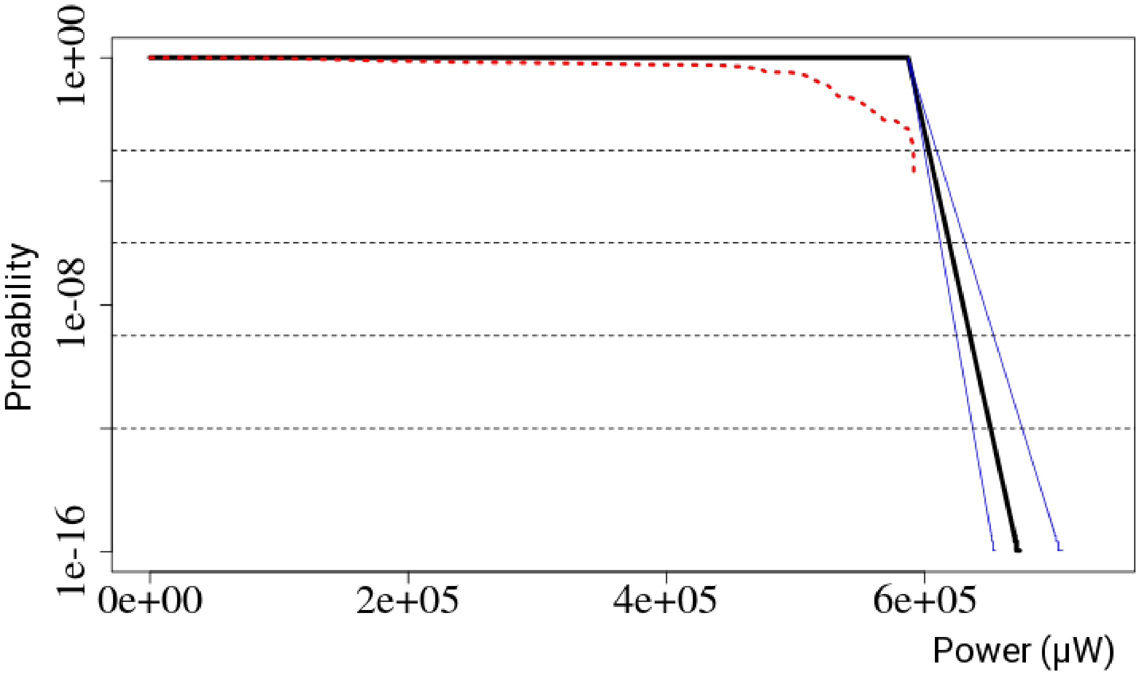


Figure 8.9: Probabilistic curve and empirical sample of power dissipation values (in μW) for benchmark `aifirf`.

estimating the number of peaks causing a voltage droop, so we focus on the latter due to limited space. For the sake of illustration, we set the threshold to determine a high power peak for samples above 95% of the maximum observed power in the deterministic setup. Table 8.1 shows how many peaks we observe in one run (execution) on the deterministic setup, the highest number of peaks per run observed across 1000 runs in the time-randomized setup and the number of peaks that could only be exceeded up to once every 10^{12} program runs[1]. The latter is derived using the number of peaks per run in the time-randomized setup as input for MBPTA-CV. As shown, the use of time-randomized setups allows us estimating the highest number of peaks expected, which ranges between few tens and few thousands of peaks. Then, by multiplying those peaks by the cost to recover from a voltage droop, the Probabilistic Worst-Case Execution Time (pWCET) estimate can be padded conveniently to account for the cost of those voltage droops. Instead, the number of peaks for the time-deterministic setup comes without any guidance on how to determine whether a higher number of peaks is possible. In fact, the deterministic nature of such a setup could lead to arbitrarily higher power peaks due to events aligning systematically.

8.5 Related Work

Power simulators have been used to provide power estimates despite the inaccuracies of their estimates, since they have been proven useful to evaluate the practicality of new techniques and perform comparisons [188, 164]. In our case, we rely on a particular simulator as a research vehicle to illustrate the applicability of our approach. However, our proposal is orthogonal to the source of the power measurements.

The use of EVT for power analysis has also being explored in [174]. In particular, this work targets maximum circuit power, for which worst-case scenarios can be created with appropriate power viruses. However, such a solution is not enough to estimate the highest power peak of a task since there is no way to relate testing data with operation behavior, and thus cannot be used for the problem considered in our work.

Resonant supply noise has also been deeply studied. Authors in [80] evaluate the events producing dangerous power peaks in a multi-core, thus allowing to improve chip-wide strategies to power-up/use cores. Some authors solve the resonant supply noise problem that these power peaks cause by using a staggered core activation [189], whereas other works suppress such supply noise by using active damping circuits [190]. In any case, those works cannot be used to forecast neither the frequency nor the magnitude of power peaks caused by user tasks, as our proposal does.

8.6 Summary

Power verification of embedded critical (real-time) systems is a mandatory step to assess their correct operation. Voltage droops caused by power peaks may lead to

[1] Other values (e.g., 10^{-9}) deliver similar conclusions.

Table 8.1: Maximum peak count for the deterministic and randomized architectures, and probabilistically estimated number of power peaks

EEMBC	**MAX Det**	**MAX Rand**	**Worst Case Number of Peaks** (10^{-12})
a2time	105	113	130
aifftr	148	183	301
aifirf	34	41	70
aiifft	142	181	281
basefp	135	148	170
bitmnp	56	60	90
cacheb	2850	2875	3042
canrdr	69	74	108
idctrn	10	21	52
iirflt	5	8	13
matrix	848	855	1102
pntrch	262	392	569
puwmod	483	489	509
rspeed	99	102	103
tblook	81	90	121
ttsprk	320	362	492

performance losses to allow recovering from those droops. Unfortunately, to the best of our knowledge, there is no practical way to estimate reliably how many such power peaks can occur in complex processors.

In this chapter we have presented an approach that, based on the use of time-randomized platforms, allows exposing power peaks during testing, breaking systematic behavior and estimating reliably the number of power peaks occurring during operation, so that the cost of recovery can be accounted for to prove that critical real-time tasks can execute timely.

In this chapter we evaluated the use of TRP for non-functional power analysis for Safety-Critical Systems (SCS). We demonstrated that TRP reveal and break pathological high power demand scenarios that are detrimental to the normal operation of SCS, hence proving that TRP expand their properties beyond the timing analysis domain and into the power analysis domain.

Part V

Conclusions and Future Work

Chapter 9

Conclusions and Future Work

> *"Labor omnia vicit improbus et duris urgens in rebus egestas."*
>
> — Vergil

Thorough Validation and Verification (V&V) of non-functional properties is a fundamental step in the development of Critical-Real Time Embedded Systems (CRTES) that differentiates them from other, more traditional, computing environments. Until recently, the most important non-functional property has been *timing correctness*, which is verified by means of Timing Analysis. For years, Timing Analysis based its working principles in the ability to control the processors states and to have specific and deep knowledge of the platform and software under analysis. However, advanced-functionality applications demand high computing performance, like autonomous driving or vehicle-to-vehicle communication. This needs can be delivered by processing platforms incorporating features and improvements that greatly increase processor's and program's complexity. This puts timing analysis in a conundrum; The well studied and existing timing analysis techniques that have to provide trustworthy Worst-Case Execution Times become prohibitively costly in front of very complex software and hardware. In this context Time-Randomized Processors (TRP) and Measurement-Based Probabilistic Timing Analysis (MBPTA) appeared to provide timing analyzability and high-performance simultaneously in CRTES. Both simplify the derivation of trustworthy Worst-Case Execution Time (WCET) estimates in complex platforms.

It is also the case that advanced-functionality software (applications) and technologies bring other challenges to the CRTES domain like security threats, diminished hardware reliability and power/energy verification difficulties. All of them are additional non-functional properties that require V&V. The coverage in the literature on how to holistically address all these new challenges of CRTES is limited. In this thesis, we have advocated for the use of TRP as a baseline paradigm that can drive future CRTES to succeed in overcoming the challenge of V&V in new CRTES improving system behavior in front of all these new challenges.

We have assessed and adapted TRP to be a solution in three of the most important non-functional metrics:

- In the reliability domain, this thesis shows that TRP extend lifetime of caches in the presence of aging effects and at the same time protect processors from pathological energy consumption scenarios that may lead to voltage droops, thus causing failures (e.g., Resonant Voltage Noise).
- In the security domain, we prove that TRP successfully defend against contention based timing attacks to caches and how randomization as a concept is used in aiding to security threat mitigation.
- In the energy domain we show how estimating energy in future CRTES will be of paramount importance and that existing models are far from providing the needed guarantees for CRTES. We show how the processes and methodologies that use TRP can be extrapolated to generate trustworthy energy estimates holding for different processor instances subject to different degrees of Process Variation. Moreover, we also demonstrate that TRP can also be used for validation of extreme power dissipation events since they naturally expose and neutralize such events.

In this thesis we assessed our hypothesis with representative use cases. In terms of hardware, we model state of the art CRTES processors, like the NGMP or the LEON4, which are currently being used or expected to be deployed in future space missions, in particular by the European Space Agency (ESA). In terms of software, space use-cases like OBDP or DEBIE, deployed in space missions or widely accepted automotive benchmarks like the Automotive EEMBC benchmark suite.

We believe this thesis to be the corner stone of a new design exploration space that considers *randomization*, and in particular TRP, as a key design paradigm of future CRTES. To that end we provided evidence of such hypothesis.

9.1 Impact

The hypotheses of this thesis are a natural follow up to projects that created and solidified the idea of using time-randomized hardware to enable probabilistic timing analysis. In that regard, this thesis further grounds the idea of TRP applicability pointing out how emerging non-functional requirements can be tackled with the use of TRP while at the same time promoting and expanding the use of MBPTA as a suitable timing analysis technique.

European level projects like Probabilistically Analyzable Real-Time Systems (FP7-PROARTIS) or Probabilistic Real-Time Control of Mixed-Criticality Multi-core and Many-core Systems (FP7-PROXIMA) settled the starting point of this thesis and have been in turn benefited by the discoveries here made.

Throughout its development, the research done in this thesis created multiple opportunities for obtaining new funding and collaborations. For instance, initial contacts with one of the groups in the Instituto Madrileño de Estudios Avanzados (IMDEA) have yielded multiple collaboration opportunities in the security field, and we are currently looking into joining efforts to further improve the security capabilities of TRP. The research community also acknowledges the impact of our proposals.

For instance, researchers producing highly relevant work in the security domain like [122, 138, 191] base their proposals around our randomized designs.

Similarly, the discoveries in this thesis lead to new funding and the creation of new projects like H2020-UP2DATE. H2020-UP2DATE looks into the future of Over-the-Air (OTA) software updates and envisions the use of software randomization to increase security of such updates, mostly in automotive systems but also in other domains.

Another project under submission that also arose from the work in this thesis is the H2020-HARDCORE project. This project will explore the uses of hardware randomization on real state of the art hardware for CRTES and test and implement new security solutions that will mitigate a wide variety of threats.

9.2 Future Work

This thesis sets the ground for several lines of future work and unexplored ideas. This work pioneers in the application of Time-Randomized Processors to solve the challenges of other non-functional metrics outside of timing predictability: Reliability, Security and Energy. The future direction in each of these 3 areas (and others) are:

- **Reliability:** Reliability and resilience is left unexplored for other resources of the processor other than the cache. For instance, the impact of aging in the TLBs or more complex cache hierarchies could also be assessed. Additionally, also other aging effects, like electromigration and the suitability of TRP to resist such events are left as future work.
- **Security:** Security is a very active field of research that is constantly spawning new threats and defenses. In this thesis, we proposed to solve conflict-based Cache-Timing Side-Channel Attacks (SCA) with an existing Time-Analyzable solution. The most immediate step would be to focus on tackling the other part of the equation, which are reuse-based SCA. These attacks rely on the fundamental working principles of the caches (temporal and spatial locality) and exploit the fact that reused secret data will exhibit shorter memory access times in contrast to the conflict-based SCA which exploits the interference patterns of cache evictions. Other approaches yet to discover might be to start from a current secure solution and make it MBPTA-compliant. At the moment it is not clear whether cache placement implementations like Random Fill [89] would suffice as they are for timing analysis. We also left the assessment of how TRP can help in other types of Side-Channel Attacks like power-based attacks as future work. In that regard, the project H2020-HARDCORE aims towards increasing the Technology Readiness Level (TRL) of security solutions presented in this work for easing the technology adoption.
- **Energy/Power:** Worst-case energy estimation is one of the most difficult and yet to be researched areas in non-functional metrics. Providing guarantees over physical phenomena that present so much variability is a challenge that will require many years to solve. Among the problems to be addressed are

mechanisms to address the estimation and accountability of the activity factor or to control and guarantee tight estimates under the presence of voltage and temperature variations. Accounting for aging variations should also be explored since it is also another source of uncertainty for energy models.

- **Temperature:** Another key non-functional metric to study that has not been addressed in this thesis is processor temperature and thermal accounting. This non-functional metric is highly related to power and energy consumption, but with different implications and solutions (e.g., power is an instantaneous issue whereas temperature changes much slower).
- **General:** Scaling the solution towards arbitrarily complex processors is also a task that involves extending the work done in this thesis. This might require exploring solutions for more complex processors that might include large networked multicores, GPUs and accelerators. The solutions proposed in this thesis set the initial path toward more mature solutions for complex architectures.

Bibliography

[1] MarketWatch. Embedded Systems Market size is growing at 5.6million USD in 2024. https://www.marketwatch.com/press-release/embedded-systems-market-size-is-growing-at-56-cagr-to-reach-95400-million-usd-in-2024-2019-02-25, February 2019. MarketWatch Press Release. 3

[2] Irune Agirre Troncoso. *Development and Certification of Mixed-criticality Embedded Systems based on Probabilistic Timing Analysis.* PhD thesis, Universitat Politècnica de Catalunya, May 2018. 3

[3] International Organization for Standardization. ISO/DIS 26262. Road Vehicles - Functional Safety, 2009. 4, 24, 45

[4] RTCA and EUROCAE. DO-178C, Software Considerations in Airborne Systems and Equipment Certification, 2012. 4, 45

[5] Martin Croxford and James Sutton. Breaking Through the V and V Bottleneck. In *International Europspace-Ada-Europe Symposium*, pages 344 – 354, 1996. 4, 8

[6] K. Hoyme and Kevin Driscoll. SAFEbus. In *Digital Avionics Systems Conference (DASC)*. IEEE, October 1992. 4

[7] Paul J. Prisaznuk. Integrated modular avionics. In *National Aerospace and Electronics Conference (NAECON)*. IEEE, May 1992. 4

[8] Christopher B. Watkins and Randy Walter. Transitioning from Federated Avionics Architectures to Integrated Modular Avionics. In *Digital Avionics Systems Conference (DASC)*. IEEE, October 2007. 4

[9] Evan Ackerman and Erico Guizzo. Ford Self-Driving Vans Will Use Legged Robots to Make Deliveries. https://spectrum.ieee.org/automaton/robotics/humanoids/ford-self-driving-vans-will-use-legged-robots-to-make-deliveries, IEEE Spectrum, May 2019. 4

[10] Mark Harris. Mentor Graphics Moves Into Automated Driving. https://spectrum.ieee.org/cars-that-think/computing/embedded-systems/mentor-graphics-moves-into-automated-driving, IEEE Spectrum, April 2017. 4

[11] ARM. ARM Expects Vehicle Compute Performance to Increase 100x in Next Decade. https://www.arm.com/company/news/2015/04/arm-expects-vehicle-compute-performance-to-increase-100x-in-next-decade, April 2015. ARM Press Release. 4

[12] Franck Wartel, Leonidas Kosmidis, Code Lo, Benoit Triquet, Eduardo Quiñones, Jaume Abella, Adriana Gogonel, Andrea Baldovin, Enrico Mezzetti, Liliana Cucu-Grosjean, and Tullio Vardanega. Measurement-Based Probabilistic Timing Analysis: Lessons from an Integrated-Modular Avionics Case Study. In *International Symposium on Industrial Embedded Systems (SIES)*. IEEE, September 2013. 5

[13] Jaume Abella, Carles Hernández, Eduardo Quiñones, Francisco J. Cazorla, Philippa Ryan Conmy, Mikel Azkarate-askasua, Jon Perez, Enrico Mezzetti, and Tullio Vardanega. WCET analysis methods: Pitfalls and challenges on their trustworthiness. In *International Symposium on Industrial Embedded Systems (SIES)*. IEEE, 2015. 6, 10, 11, 19, 22, 64

[14] Ann Steffora Mutschler. Transistor Aging Intensifies At 10/7nm And Below. https://semiengineering.com/transistor-aging-intensifies-10nm/, July 2017. 6

[15] Brian Bailey. Chip Aging Becomes Design Problem. https://semiengineering.com/chip-aging-becomes-design-problem/, August 2018. 6

[16] Shekhar Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *MICRO*, pages 10–16, September 2005. 6

[17] Steve Guertin and Mark White. CMOS Reliability Challenges - The Future of Commercial Digital Electronics and NASA. In *NEPP Electronic Technology Workshop*, June 2010. 6

[18] General Electric. Mil/Aero Thermal Management. Technical report, 2013. 6, 8

[19] Joseph H. Saleh, Daniel E. Hastings, and Dava J. Newman. Spacecraft design lifetime. *Journal of Spacecraft and Rockets (JSR)*, 39, March 2002. 6

[20] Damien Hardy and Isabelle Puaut. Static Probabilistic Worst Case Execution Time Estimation for Architectures with Faulty Instruction Caches. In *International conference on Real-Time Networks and Systems (RTNS)*. ACM, October 2013. 6, 60

[21] Damien Hardy, Isabelle Puaut, and Yiannakis Sazeides. Probabilistic WCET Estimation in Presence of Hardware for Mitigating the Impact of Permanent Faults. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. EDA Consortium, March 2016. 6, 60

[22] Jaume Abella, Eduardo Quiñones, Francisco J. Cazorla, Yiannakis Sazeides, and Mateo Valero. RVC: A Mechanism for Time-Analyzable Real-Time Processors with Faulty Caches. In *International Conference on High Performance and Embedded Architectures and Compilers (HiPEAC)*. ACM, Jaunary 2011. 6, 60

[23] Mladen Slijepcevic, Leonidas Kosmidis, Jaume Abella, Eduardo Quiñones, and Fracisco J. Cazorla. Timing Verification of Fault-Tolerant Chips for Safety-Critical Applications in Harsh Environments. *Micro*, 34(6):8–19, December 2014. 6, 45, 60

[24] The Council of Economic Advisers. The Cost of Malicious Cyber Activity to the U.S. Economy. Technical report, February 2018. 7

[25] Augusto Vega, Alper Buyuktosunoglu, and Pradip Bose. Towards "Smarter" Vehicles Through Cloud-Backed Swarm Cognition. In *Intelligent Vehicles Symposium (IV)*. IEEE, June 2018. 7

[26] Akin Sisbot, Augusto Vega, Arun Paidmarri, John-David Wellman, Alper Buyuktosunoglu, Pradip Bose, and David Trilla. Multi-Vehicle Map Fusion Using Gnu Radio. In *GNU Radio Conference (GRCON)*, September 2019. 7

[27] Dennis K. Nilsson and Ulf E. Larson. Secure Firmware Updates over the Air in Intelligent Vehicles. In *International Conference on Communications Workshop (ICCW)*, May 2008. 7, 31

[28] The Advanced Rechargeable & Lithium Batteries Association. The Batteries Report. Technical report, April 2018. 7

[29] U.S. Department Of Health, Human Services Food, and Drug Administration. General Principles of Software Validation; Final Guidance for Industry and FDA Staff. `https://www.fda.gov/media/73141/download`, January 2012. 8

[30] Curtiss-Wright. Understanding Power Management of Intel Processors for Mil/Aero Applications. Technical report, 2011. 8

[31] Leonidas Kosmidis, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. A Cache Design for Probabilistically Analysable Real-Time Systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2013. 9, 25, 28, 45, 60, 65, 67, 68

[32] Fracisco J. Cazorla, Leonidas Kosmidis, Enrico Mezzetti, Carles Hernández, Jaume Abella, and Tullio Vardanega. Probabilistic Worst-Case Timing Analysis: Taxonomy and Comprehensive Survey. *ACM Computing Surveys*, 52, 2019. 9, 18, 20, 22, 24, 25

[33] Jaume Abella, Maria Padilla, Joan Del Castillo, and Francisco J. Cazorla. Measurement-Based Worst-Case Execution Time Estimation Using the Coefficient of Variation. *Transactions on Design Automation of Electronic Systems (TODAES)*, 22(4):72:1–72:29, July 2017. 9, 10, 21, 22, 38, 102

[34] Liliana Cucu-Grosjean, Luca Santinelli, Michael Houston, Code Lo, Tullio Vardanega, Jaume Abella, Enrico Mezzetti, Eduardo Quiñones, and Francisco J. Cazorla. Measurement-Based Probabilistic Timing Analysis for Multi-path Programs. In *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, July 2012. 10, 20, 21, 22, 45, 64, 65

[35] Carles Hernández, Jaume Abella, Francisco J. Cazorla, Alen Bardizbanyan, Jan Andersson, Fabrice Cros, and Franck Wartel. Design and Implementation of a Time Predictable Processor: Evaluation With a Space Case Study. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 16:1–16:23, 2017. 10, 25, 75

[36] Enrico Mezzetti, Marco Ziccardi, Tullio Vardanega, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. Randomized Caches Can Be Pretty Useful to Hard Real-Time Systems. *Leibniz Transactions on Embedded Systems (LITES)*, 2(1), July 2015. 10, 25

[37] Carles Hernández, Jaume Abella, Andrea Gianarro, Jan Andersson, and Francisco J. Cazorla. Random Modulo: A New Processor Cache Design for Real-Time Critical Systems. In *Design Automation Conference (DAC)*. IEEE, 2016. 12, 25, 26, 27, 28, 45, 59, 60, 64, 65, 67, 68, 74, 117

[38] Cheng T. Wang. *Hot Carrier Design Considerations for MOS Devices and Circuits*. Van Nostrand Reinhold, 1992. 12, 30

[39] Dieter K. Schroder and Jeff A. Babcock. Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing. *Journal of Applied Physics*, 94(1):1–18, 2003. 12, 30

[40] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware. *Journal of Cryptographic Engineering*, 8:1–27, October 2016. 12, 76

[41] Daniel J. Bernstein. Cache-Timing Attacks on AES, April 2005. 12, 32, 66, 71

[42] Robert Ian Davis and Liliana Cucu-Grosjean. A Survey of Probabilistic Schedulability Analysis Techniques for Real-Time Systems. pages 1–53, May 2019. 18, 20

[43] Sudipta Chattopadhyay, Chong Lee Kee, Abhik Roychoudhury, Timon Kelter, Peter Marwedel, and Heiko Falk. A Unified WCET Analysis Framework for Multi-core Platforms. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, April 2012. 18

[44] Yau-Tsun Steven Li and Sharad Malik. Performance Analysis of Embedded Software Using Implicit Path Enumeration. *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCADICS)*, 16:1477–1487, 1997. 19

[45] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. The Worst-Case Execution-Time Problem: Overview of Methods and Survey of Tools. *Transactions on Embedded Computing Systems (TECS)*, 7:1–53, May. 19, 20

[46] I. Wenzel. *Measurement-Based Timing Analysis of Superscalar Processors*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, 2006. 20

[47] Francisco J. Cazorla, Eduardo Quiñones, Tullio Vardanega, Liliana Cucu-Grosjean, Benoit Triquet, Guillem Bernat, Emery Berger, Juame Abella, Franck Wartel, Michael Houston, Luca Santinelli, Leonidas Kosmidis, Code Lo, and Dorin Maxim. PROARTIS: Probabilistically Analyzable Real-Time Systems. *Transaction on Embedded Computing Systems (TECS)*, 12(94), May 2013. 20, 28

[48] Francisco J. Cazorla, Tullio Vardanega, Eduardo Quiñones, and Jaume Abella. Upper-bounding Program Execution Time with Extreme Value Theory. In *Workshop on Worst-Case Execution Time Analysis (WCET)*, July 2013. 20

[49] Leonidas Kosmidis, Jaume Abella, Franck Wartel, Eduardo Quiñones, Antoine Colin, and Fracisco J. Cazorla. PUB: Path Upper-Bounding for Measurement-Based Probabilistic Timing Analysis. In *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, August 2014. 21, 90, 92

[50] Karila Palma Silva, Luis Fernando Arcaro, and Romulo SIlva de Oliveira. On Using GEV or Gumbel Models when Applying EVT for Probabilistic WCET Estimation. In *Real-Time Systems Symposium (RTSS)*. IEEE, December 2017. 21, 22, 102, 110

[51] George Lima, Dário Dias, and Edna Barros. Extreme Value Theory for Estimating Task Execution Time Bounds: A Careful Look. In *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, July 2016. 21, 102, 116

[52] Suzana Milutinovic, Jaume Abella, Enrico Mezzetti, and Francisco J. Cazorla. Measurement-based Cache Representativeness on Multipath Programs. In *Design Automation Conference (DAC)*, June 2018. 21, 25

[53] Suzana Milutinovic, Enrico Mezzetti, Jaume Abella, and Francisco J. Cazorla. Increasing the reliability of software timing analysis for cache-based processors. *Transactions on Computers (TC)*, January 2019. 21, 25

[54] Mikel Fernández, Roberto Gioiosa, Eduardo Quiñones, Luca Fossati, Marco Zulianello, and Francisco J. Cazorla. Assessing the Suitability of the NGMP Multi-Core Processor in the Space Domain. In *International Conference on Embedded Software (EMSOFT)*. ACM, October 2012. 21

[55] Samuel Kotz and Saralees Nadarajah. *Extreme Value Distributions: Theory and Applications*. World Scientific, October 2000. 22, 99, 102, 110

[56] Suzana Milutinovic, Enrico Mezzetti, Jaume Abella, Tullio Vardanega, and Francisco J. Cazorla. On Uses of Extreme Value Theory Fit for Industrial-Quality WCET Analysis. In *International Symposium on Industrial Embedded Systems (SIES)*. IEEE, June 2017. 22, 103

[57] William Feller. *An introduction to Probability Theory and Its Applications*. January 1968. 23, 74, 102

[58] G.E.P. Box and David A. Pierce. Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models. *Journal of the American Statistical Association*, 65(332):1509–1526, December 1970. 23, 74, 102

[59] Franck Wartel, Leonidas Kosmidis, Adriana Gogonel, Andrea Baldovino, Zoe Stephenson, Benoit Triquet, Eduardo Quiñones, Code Lo, Enrico Mezzetti, Ian Broster, Jaume Abella, Liliana Cucu-Grosjean, Tullio Vardanega, and Francisco J. Cazorla. Timing Analysis of an Avionics Case Study on Complex Hardware/Software Platforms. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2015. 23

[60] Mikel Fernández, David Morales, Leonidas Kosmidis, Alen Bardizbanyan, Ian Broster, Carles Hernández, Eduardo Quiñones, Jaume Abella, Francisco J. Cazorla, Paulo Machado, and Luca Fossati. Probabilistic Timing Analysis on Time-Randomized Platforms for the Space Domain. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, March 2017. 23

[61] Leonidas Kosmidis, Davide Compagnin, David Morales, Enrico Mezzetti, Eduardo Quinones, Jaume Abella, Tullio Vardanega, and Francisco J. Cazorla. Measurement-Based Timing Analysis of the AURIX Caches. In Martin Schoeberl, editor, *16th International Workshop on Worst-Case Execution Time Analysis (WCET 2016)*, volume 55, pages 9:1–9:11, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 23

[62] Miloš Panić, Carles Hernandez, Jaume Abella, Antoni Roca, Eduardo Quiñones, and Francisco J. Cazorla. Improving Performance Guarantees in Wormhole Mesh NoC Designs. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, March 2016. 24

[63] Fracisco J. Cazorla, Jaume Abella, Enrico Mezzetti, Carles Hernandez, Tullio Vardanega, and Guillem Bernat. Reconciling Time Predictability and Performance in Future Computing Systems. *Design & Test (D&T)*, pages 48–56, April 2018. 24

[64] Leonidas Kosmidis, Charlie Curtsinger, Eduardo Quiñones, Jaume Abella, Emery Berger, and Francisco J. Cazorla. Probabilistic timing analysis on conventional cache designs. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2013. 24, 25

[65] Leonidas Kosmidis, Eduardo Quiñones, Jaume Abella, Glenn Farrall, Franck Wartel, and Fracisco J. Cazorla. Containing Timing-Related Certification Cost in Automotive SYstems Deploying Complex Hardware. In *Design Automation Conference (DAC)*. IEEE, 2014. 24, 25

[66] Leonidas Kosmidis, Roberto Vargas, David Morales, Eduardo Quiñones, Jaume Abella, and Francisco J. Cazorla. TASA: Toolchain Agnostic Software Randomisation for Critical Real-Time Systems. In *International Conference on Computer-Aided Design (ICCAD)*. IEEE, November 2016. 25

[67] Eduardo Quiñones, Emergy D. Berger, Guillem Bernat, and Francisco J. Cazorla. Using Randomized Caches in Probabilistic Real-Time Systems. In *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, July 2009. 25, 28

[68] Jaume Abella, Eduardo Quiñones, Franck Wartel, Tullio Vardanega, and Francisco J. Cazorla. Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA. In *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, July 2014. 25

[69] François Bodin and André Seznec. Skewed Associativity Improves Program Performance and Enhances Predictability. *Transactions on Computers (TC)*, 46:530–544, May 1997. 25

[70] Michael S. Schlansker, Robert L. Shaw, and S. Sivaramakrishnan. Randomization and Associativity in the Design of Placement-Insensitive Caches. Technical report, 1993. 25

[71] Leonidas Kosmidis, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. Efficient Cache Designs for Probabilistically Analysable Real-Time Systems. *Transactions on Computers (TC)*, 63(12):2998–3011, September 2014. 25, 50, 60

[72] Carles Hernández, Jaume Abella, Fracisco J. Cazorla, Jan Andresson, and Andrea Gianarro. Towards Making a LEON3 Multicore Compatible with Probabilistic Timing Analysis. In *Data Systems in Aerospace (DASIA)*, May 2015. 25

[73] Pedro Benedicte, Carles Hernández, Jaume Abella, and Fracisco J. Cazorla. Design and Integration of Hierarchical-Placement Multi-Level Caches for Real-Time Systems. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018. 28

[74] Sebastian Altmeyer, Liliana Cucu-Grosjean, and Robert Ian Davis. Static Probabilistic Timing Analysis for Real-Time Systems Using Random Replacement Caches. *Journal of Real-Time Systems (RTS)*, January 2015. 28

[75] Pedro Benedicte, Carles Hernández, Jaume Abella, and Francisco J. Cazorla. Rpr: A random replacement policy with limited pathological replacements. In *Symposium on Applied Computing (SAC)*. ACM, April 2018. 28

[76] Javier Jalle, Leonidas Kosmidis, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. Bus Designs for Time-Probabilistic Multicore Processors. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014. 29

[77] Kanishka Lahiri, Anand Raghunathan, and Ganesh Lakshminarayana. The LOTTERYBUS On-Chip Communication Architecture. *Transactions on Very Large Scale Integration (VLSI) Systems*, pages 596–608, June 2006. 29

[78] Mladen Slijepcevic, Mikel Fernández, Carles Hernández, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. pTNoC: Probabilistically Time-Analyzable Tree-Based NoC for Mixed-Criticality Systems. In *Euromicro Conference on Digital System Design*. IEEE, August 2016. 29

[79] Muhammad A. Alam. A critical examination of the mechanics of dynamic NBTI for PMOSFETs. In *International Electron Devices Meeting (IEDM)*. IEEE, 2003. 30, 60

[80] Ramon Bertran, Alper Buyuktosunoglu, Pradip Bose, Timothy J. Slegel, Gerard Salem, Sean Carey, Richard F. Rizzolo, and Thomas Strach. Voltage Noise in Multi-Core Processors: Empirical Characterization and Optimization Opportunities. In *International Symposium on Microarchitecture (MICRO)*. IEEE, December 2014. 31, 33, 60, 90, 92, 119

[81] Jakub Szefer. Survey of Microarchtiectural Side and Covert Channels, Attacks and Defenses. *Journal of Hardware and Systems Security*, September 2016. 31, 66, 76

[82] Jun Xu, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. Transparent Runtime Randomization for Security. pages 260–269, 2003. 31, 75

[83] Thomas Moscibroda and Onur Mutlu. Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems. In *USENIX Security Symposium (SS)*, August 2007. 31

[84] Nathan P. Smith. Stack Smashing Vulnerabilities in the UNIX Operating System, 1997. 31

[85] Ari Takanen, Marko Laakso, Juhani Eronen, and Juha Röning. Running Malicious Code By Exploiting Buffer Overflows: A Survey Of Publicly Available Exploits. In *European Institute for Computer Anti-VIrus Research (EICAR) Conference*, 2000. 31

[86] Prathap Kumar Valsan, Heechul Yun, and Farzad Farshchi. Taming Non-Blocking Caches to Improve Isolation in Multicore Real-Time Systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, April 2016. 32

[87] Joseph Bonneau and Ilya Mironov. Cache-Collision Timing Attacks Against AES. In *International conference on Cryptographic Hardware and Embedded Systems (CHES)*. Springer-Verlag, October 2006. 32

[88] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptoanalysis of DES Implemented on Computers with Cache. In *International conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2003. 32

[89] Fangfei Liu and Ruby B. Lee. Random Fill Cache Architecture. In *International Symposium on Microarchitecture (MICRO)*. IEEE, December 2014. 32, 76, 125

[90] J. Adam Butts and Gurindar S. Sohi. A Static Power Model for Architects. In *International symposium on Microarchitecture (MICRO)*. ACM, 2000. 33

[91] Krisztián Flautner, Nam Sung Kim, Steve Martin, David Blaauw, and Trevor Mudge. Drowsy Caches: Simple Techniques for Reducing Leakage Power. In *Annual International Symposium on Computer Architecture (ISCA)*. IEEE, May 2002. 33

[92] Enrico Macii, Massoud Pedram, and Fabio Somenzi. High-Level Power Modeling, Estimation, and Optimization. *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 17(11):1061–1079, November 1998. 34

[93] Cobham Gaisler. LEON4 Processor. `https://www.gaisler.com/index.php/products/processors/leon4`, June 2008. 35

[94] Cobham Gaisler. Quad Core LEON4 SPARC V8 Processor LEON4-NGMP-DRAFT Data Sheet and User's Manual. `http://microelectronics.esa.int/gr740/LEON4-NGMP-DRAFT-2-1.pdf`, 2013. 35, 36, 51, 57, 117

[95] NXP. *e200z4 Power Architecture Core Reference Manual*, 2009. `https://www.nxp.com/docs/en/reference-manual/e200z4RM.pdf`. 35, 36, 71

[96] Cobham Gaisler. LEON3 Processor. https://www.gaisler.com/index.php/products/processors/leon3, June 2008. 35

[97] Inc. SPARC International. *The SPARC Architecture Manual – Version 8*, 1992. https://www.gaisler.com/doc/sparcv8.pdf. 36

[98] Cobham Gasiler. *GR-CPCI-GR740 Development Board User's Manual*, 2019. 36

[99] LiP6. SoCLib. http://www.soclib.fr/trac/dev, 2011. SoCLiB Webpage. 38, 51, 71, 103

[100] Javier Jalle, Jaume Abella, Luca Fossati, Marco Zulianello, and Francisco J. Cazorla. Validating a Timing Simulator for the NGMP Multicore Processor. In *Data Systems in Aerospace (DASIA)*, May 2016. 38

[101] Sheng Li, Jun Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *International Symposium on Microarchitecture (MICRO)*, December 2009. 38, 92, 104

[102] Synopsys. Synopsys Design Compiler. http://www.synopsys.com/. 38, 52

[103] TSMC. TSMC 40 nm technology. http://www.tsmc.com/english/dedicatedFoundry/technology/40nm.htm. 38, 52

[104] Mentor. Questa advanced simulator. https://www.mentor.com/products/fv/questa/. 38

[105] Jaume Abella. MBPTA-CV. https://doi.org/10.5281/zenodo.1065776, November 2017. 38

[106] Jason A. Poovey, Thomas M. Conte, Markus Levy, and Shay Gal-On. A Benchmark Characterization of the EEMBC Benchmark Suite. *MICRO*, pages 18–29, September 2009. 38, 39, 51, 104, 117

[107] J. P. Schwanethal, N. McBride, S. F. Green, J. A. M. McDonnell, and G. DDrolshagen. April 2005. 39

[108] European Space Agency. On-Board Data Processing – Benchmarks. http://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Data_Processing/General_Benchmarking_and_Specific_Algorithms, October 2015. 39

[109] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper. The Mälardalen WCET Benchmarks: Past, Present and Future. In *Workshop on Worst-Case Execution Time Analysis (WCET)*, pages 136–146, July 2010. 39

[110] Leonidas Kosmidis, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. Multi-Level Unified Caches for Probabilistically Time Analysable Real-Time Systems. In *Real-Time Systems Symposium (RTSS)*. IEEE, December 2013. 45, 60

[111] Mladen Slijepcevic, Leonidas Kosmidis, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. DTM: Degraded Test Mode for Fault-Aware Probabilistic Timing Analysis. In *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, July 2013. 45, 60

[112] Marco Ziccardi, Enrico Mezzetti, Tullio Vardanega, Jaume Abella, and Francisco J. Cazorla. EPC: Extended Path Coverage for Measurement-based Probabilistic Timing Analysis. In *Real-Time Systems Symposium (RTSS)*. IEEE, December 2015. 51

[113] Hyungjun Kim, Siva Bhanu Krishna Boga, Arsenjy Vitkovskiy, Stavros Hadjitheophanous, Paul V. Gratz, Vassos Soteriou, and Maria K. Michael. Use It or Lose It: Proactive, Deterministic Longevity in Future Chip Multiprocessors. *Transactions on Design Automation of Electronic Systems (TODAES)*, 20(4):65:1–65:26, September 2015. 51, 55, 60

[114] Erika Gunadi, Abhisek A. Sinkar, Nam Sung Kim, and Mikko H. Lipasti. Combating Aging with the Colt Duty Cycle Equalizer. In *International Symposium on Microarchitecture (MICRO)*. IEEE, December 2010. 58, 60

[115] Sanjay V. Kumar, Chris H. Kim, and Sachin S. Sapatnekar. Impact of NBTI on SRAM read stability and design for reliability. In *International Symposium on Quality Electronic Design (ISQED)*. IEEE, March 2006. 58, 60

[116] Jaume Abella, Xavier Vera, and Antonio González. Penelope: The NBTI-Aware Processor. In *International Symposium on Microarchitecture (MICRO)*. IEEE, December 2007. 58, 60

[117] Jeonghee Shin, Victor Zyuban, Pradip Bose, and Timothy M. Pinkston. A Proactive Wearout Recovery Approach for Exploiting Microarchitectural Redundancy to Extend Cache SRAM Lifetime. In *International Symposium on Computer Architecture (ISCA)*. IEEE, June 2008. 60

[118] Vincent Huard, Remy Chevallier, Chittoor Parthasarathy, Anand Mishra, Natalia Ruiz-Amador, Flore Persin, Vincent Robert, Alejandro Chimeno, Emmanuel Pion, Nicolas Planes, David Ney, Florian Cachoe, Neeraj Kapoor, Vishal Kulshrestha, Sanjeev Chopra, and Nicolas Vialle. Managing SRAM Reliability from Bitcell to Library Level. In *International Reliability Physics Symposium (IRPS)*. IEEE, May 2010. 60

[119] J. Zhang, A. Marathe, K. Taylor, E. Zhao, and B. En. New Findings of NBTI in Partially Depleted SOI Transistors with Ultra-Thin Gate Dielectrics. In

International Reliability Physics Symposium. Proceedings (IRPS). IEEE, April 2004. 60

[120] Jaume Abella, Eduardo Quiñones, Francisco J. Cazorla, Mateo Valero, and Yanos Sazeides. RVC-Based Time-Predictable Faulty Caches for Safety-Critical Systems. In *International On-Line Testing Symposium (IOLTS)*. IEEE, July 2011. 60

[121] Damien Hardy and Isabelle Puaut. Static Probabilistic Worst Case Execution Time Estimation for Architectures with Faulty Instruction Caches. *Journal of Real-Time Systems (RTS)*, 51(2):128–152, November 2015. 60

[122] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *Symposium on Security and Privacy (S&P)*, 2019. 63, 125

[123] Enrico Mezzetti and Tullio Vardanega. A rapid cache-aware procedure positioning otimization to favor incremental development. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, April 2013. 64

[124] David Trilla, Carles Hernández, Jaume Abella, and Francisco J. Cazorla. Resilient Random Modulo Cache Memories for Probabilistically-Analyzable Real-Time Systems. In *International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, July 2016. 64, 65, 67, 68

[125] Irune Agirre, Mikel Azkarate-askasua, Carles Hernández, Jaume Abella, Jon Perez, Tullio Vardanega, and Francisco J. Cazorla. IEC-61508 SIL 3 Compliant Pseudo-Random Number Generators for Probabilistic Timing Analysis. In *Euromicro Conference on Digital System Design (DSD)*. IEEE, August 2015. 65, 67

[126] Zhenghong Wand and Ruby B. Lee. New Cache Designs for Thwarting software Cache-based Side Channel Attacks. In *International Symposium on Computer Architecture (ISCA)*, pages 494–505. ACM, June 2007. 66, 71, 76

[127] Zhenghong Wang and Ruby B. Lee. A Novel Cache Architecture with Enhanced Performance and Security. In *International Symposium on Microarchitecture (MICRO)*, pages 83–93. IEEE, 2008. 67

[128] Onur Aciicmez, Jean-Pierre Seifert, Qingwei MA, and Xinwen Zhang. Method and system for securing instruction caches using substantially random instruction mapping scheme, November 2011. US Patent 8,055,848. 67

[129] Václav Edvard Beneš. Optimal Rearrangeable Multistage Connecting Networks. *Bell System Technical Journal*, 43:1641–1656, July 1964. 68

[130] Hongfei Qu, Jinfu Xu, and Yingjian Yan. A Random Delay Design of Processor Against Power Analysis Attacks. In *International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. IEEE, November 2010. 75

[131] Charlie Curtsinger and Emery D. Berger. STABILIZER: Statistically Sound Performance Evaluation. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2013. 75

[132] Leonid Domnitser, Aamer Jaleel, Jason Loew, Nael Abu-Ghazaleh, and Dmitry Ponomarev. Non-Monopolizable Caches: Low-Complexity Mitigation of Cache Side Channel Attacks. *Transactions on Architecture and Code Optimization (TACO)*, 8:35:1–35:21, January 2012. 76

[133] Marco Paolieri, Eduardo Quiñones, Francisco J. Cazorla, Guillem Bernat, and Mateo Valero. Hardware Support for WCET Analysis of Hard Real-Time Multicore Systems. In *International symposium on Computer architecture (ISCA)*. ACM, June 2009. 76

[134] Mladen Slijepcevic, Leonidas Kosmidis, Jaume Abella, Eduardo Quiñones, and Francisco J. Cazorla. Time-Analysable Non-Partitioned Shared Caches for Real-Time Multicore Systems. In *Design Automation Conference (DAC)*. ACM, June 2014. 76

[135] Thomas Bourgeat, Ilia Lebedev, Andrew Wright, Sizhuo Zhang, Arvind, and Srinivas Devadas. Mi6: Secure enclaves in a speculative out-of-order processor. In *International Symposium on Microarchitecture (MICRO)*. ACM, October 2019. 76

[136] Moinudding K. Qureshi. CEASER: Mitigating Conflict-Based Cache Attacks via Encrypted-Address and Remapping. In *International Symposium on Microarchitecture (MICRO)*. IEEE, October 2018. 76

[137] Pepe Vila, Boris Köpf, and José Francisco Morales. Theory and Practice of Finding Eviction Sets. In *Symposium on Security and Privacy (SP)*. IEEE, May 2018. 76

[138] Mario Werner, Thomas Unterluggauer, Lukas Giner, Michael Schwarz, Daniel Gruss, and Stefan Mangrad. SCATTERCACHE: Thwarting Cache Attacks via Cache Set Randomization. In *USENIX Conference on Security (SEC)*, August 2019. 76, 125

[139] Ralph Görgen, Kim Grüttner, Fernando Herrera, Pablo Peñil, Julio Medina, Eugenio Villar, Gianluca Palermo, William Fornaciari, Carlo Brandolese, Davide Gadioli, Sara Bocchio, Luca Ceva, Paolo Azzoni, Massimo Poncino, Sara Vinco, Enrico Macii, Salvatore Cusenza, John Favaro, Raúl Valencia, Ingo Sander, Kathrin Rosvall, and Davide Quaglia. CONTREX: Design of embedded mixed-criticality CONTRol systems under consideration of EXtra-functional properties. *Microprocessors and Microsystems (MICPRO)*, 51:39–55, April 2017. 79

[140] Certification Authorities Software Team (CAST). Multi-core Processors. Technical report, CAST-32A, November 2016. 79

[141] Saumya Chandra, Kanishka Lahiri, Anand Raghunathan, and Sujit Dey. Considering Process Variations During System-Level Power Analysis. In *international Symposium on Low Power Electronics and Design (ISPLED)*. ACM, October 2006. 81, 88

[142] Shekhar Borkar, Tanay Karnik, Siva G. Narendra, James W. Tschanz, Ali Keshavarzi, and Vivek K. De. Parameter Variations and Impact on Circuits and Microarchitecture. In *Design Automation Conference (DAC)*. IEEE, June 2003. 81, 88

[143] Keith A. Bowman, Steven G. Duvall, and James D. Meindl. Impact of Die-to-Die and Within-Die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration. *Journal of Solid-State Circuits (JSSC)*, 37(2):183–190, August 2002. 81, 82, 99

[144] Jaume Abella and Xavier Vera. Electromigration for Microarchitects. *Computing Surveys (CSUR)*, 42(2):9:1–9:18, February 2010. 81

[145] Dieter Schroder. Negative Bias Temperature Instability: What Do We Understand? *Microelectronics Reliability*, 47(6):841–852, 2007. 81

[146] Kueing-Long Chen, S. A. Saller, I. S. Groves, and D. B. Scott. Reliability Effects on MOS Transistors Due to Hot-Carrier Injection. *Transactions on Electron Devices (TED)*, 32:386–393, February 1985. 81

[147] Infineon. *TC270/TC275/TC277 32-Bit Single-Chip Micocontroller*, January 2017. https://www.infineon.com/dgdl/Infineon-TC27xDC_DS_v10-DS-v01_00-EN.pdf?fileId=5546d46259d9a4bf015a846b292f74ce. 82

[148] Intel. Measuring Processor Power: TDP vs. ACP. https://www.intel.com/content/dam/doc/white-paper/resources-xeon-measuring-processor-power-paper.pdf, April 2011. Intel White Paper. 82, 98

[149] AMD. ACP – The Truth About Power Consumption Starts Here. https://www.amd.com/Documents/43761D-ACP_PowerConsumption.pdf, 2011. AMD White Paper. 82

[150] James W. Tschanz, James T. Kao, Siva G. Narendra, Raj Nair, Dimitri A. Antoniadis, Anantha P. Chandrakasan, and Vivek K. De. Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage. *Journal of Solid-State Circuits (JSSC)*, 37(11):1396–1402, November 2002. 82

[151] Pradip Bose. Pre-Silicon Modeling and Analysis: Impact On Real Design. *Micro*, 26(4):3–3, July 2006. 82

[152] Xuemei Xi, Mohan Dunga, Jin He, Weidong Liu, Kanyu M. Cao, Xiaodong Jin, Jeff J. Ou, Mansun Chan, and Ali M. Niknejad. *BSIM4.3.0 MOSFET Model - User's Manual*, 2003. http://ewh.ieee.org/r5/denver/sscs/References/2003_BSIM4v30_manual.pdf. 82

[153] Mikako Miyama, Shiro Kamohara, Mitsuru Hiraki, Kazunori Onozawa, and Hisaaki Kunitomo. Pre-Silicon Parameter Generation Methodology Using BSIM3 for Circuit Performance-Oriented Device Optimization. *Transactions on Semicondutor Manufacturing (TSM)*, 14(2):134–142, May 2001. 83

[154] Victor Jiménez, Fracisco J. Cazorla, Roberto Gioiosa, Mateo Valero, Carlos Boneti, Eren Kursun, Chen-Yong Cher, Canturk Isci, Alper Buyuktosunoglu, and Pradip Bose. Power and thermal characterization of POWER6 system. In *Parallel Architectures and Compilation Techniques (PACT)*. ACM, September 2010. 83

[155] Michael Floyd, Malcolm Ware, Karthick Rajamani, Tilman Gloekler, Bishop Brock, Pradip Bose, Alper Buyuktosunoglu, Juan C. Rubio, Birgit Schubert, Bruno Spruth, Jose A. Tierno, and Lorena Pesantez. Adaptive energy-management features of the IBM POWER7 chip. *IBM Journal of Research and Development*, 55(3), 2011. 83

[156] Karthik Ganesan, Jungho Jo, W. Lloyd Bircher, Dimitris Kaseridis, Zhibin Yu, and Lizy K. John. System-level Max POwer (SYMPO) - a Systematic Approach for Escalating System-Level Power Consumption Using Synthetic Benchmarks. In *Parallel Architectures and Compilation Techniques (PACT)*, September 2010. 84, 85, 110

[157] EETimes. Intel cancels Tejas, moves to dual-core designs. https://www.eetimes.com/document.asp?doc_id=1150169, May 2004. 84

[158] Ramkumar Jayaseelan, Tulika Mitra, and Xianfeng Li. Estimating the Worst-Case Energy Consumption of Embedded Software. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, April 2006. 84, 93

[159] Peter Wägemann, Tobias Distler, Timo Hönig, Heiko Janker, Rüdiger Kapitza, and WOlfgang Schröder-Preikschat. Worst-Case Energy Consumption Analysis for Energy-Constrained Embedded Systems. In *Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, July 2015. 84, 93

[160] Deo Singh, Jan M. Rabaey, Massoud Pedram, Francky Catthoor, Suresh Rajgopal, Neelima Sehgal, and Thomas J. Mozdzen. Power conscious CAD tools and methodologies: a perspective. *Proceedings of the IEEE*, 83(4):570–594, April 1995. 86

[161] Trent McConaghy. Analog Behavior in Custom IC Variation-Aware Design. In *International Conference on Computer-Aided Design (ICCAD)*, pages 146–148, November 2013. 87

[162] EECS Department of the University of California at Berkeley. Spice. www.http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/, 2001. SPICE Webpage. 92

[163] Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P. Jouppi. *CACTI 5.1*, April 2008. https://www.hpl.hp.com/techreports/2008/HPL-2008-20.html. 92

[164] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a Framework for Architectural-Level Power Analysis and Optimizations. In *International Symposium on Computer Architecture (ISCA)*. ACM, May 2000. 92, 119

[165] Bhavishya Goel and Sally A. McKee. A Methodology for Modeling Dynamic and Static Power Consumption for Multicore Processors. In *International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, May 2016. 92

[166] Kristoffer Robin Stokke, Håkon Kvale Stensland, Pål Halvorsen, and Carsten Griwodz. High-Precision Power Modelling of the Tegra K1 Variable SMP Processor Architecture. In *International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*. IEEE, September 2016. 92

[167] Sriram Sankaran. Predictive Modeling Based Power Estimation for Embedded Multicore Systems. In *nternational Conference on Computing Frontiers (CF)*, May 2016. 92

[168] Sam Van den Steen, Sander De Pestel, Moncef Mechri, Stijn Eyerman, Trevor Carlson, David Black-Schaffer, Erick Hagersten, and Lieven Eeckhout. Micro-Architecture Independent Analytical Processor Performance and Power Modeling. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, March 2015. 92

[169] Waltenegus Dargie. A Stochastic Model for Estimating the Power Consumption of a Processor. *Transactions on Computers*, pages 1311–1322, April 2014. 92

[170] Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. A validation of DRAM RAPL Power Measurements. In *International Symposium on Memory Systems (MEMSYS)*. IEEE, October 2016. 92

[171] Jóakim von Kistowski, Hansfried Block, John Beckett, Cloyce Spradling, Klaus-Dieter Lange, and Samuel Kounev. Variations in CPU Power Consumption. In *International Conference on Performance Engineering (ICPE)*, pages 147–158. ACM, marh 2016. 92

[172] James Pallister, Steve Kerrison, Jeremy Morse, and Kerstin Eder. Data Dependent Energy Modeling for Worst Case Energy Consumption Analysis. In *International Workshop on Software and Compilers for Embedded Systems (SCOPES)*, June 2017. 92

[173] Jeremy Morse, Steve Kerrison, and Kerstin Eder. On the Limitations of Analyzing Worst-Case Dynamic Energy of Processing. *Transactions on Embedded Computing Systems (TECS)*, 17:59:1–59:22, June 2018. 92

[174] Nestoras E. Evmorfopoulos, Giorgios I. Stamoulis, and John N. Avaritsiotis and. A Monte Carlo Approach for Maximum Power Estimation Based on Extreme Value Theory. *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 21:415–432, 2002. 93, 119

[175] Peter Wägemann, Christian Dietrich, Tobias Distler, Peter Ulbrich, and Wolfgang Schröder-Preikschat. Whole-System WCEC Analysis for Energy-Constrained Real-Time Systems. In *Euromicro Conference on Real-Time Systems (ECRTS)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. 93

[176] Carles Hernández, Federico Silla, and José Duato. A Methodology for the Characterization of Process Variation in NoC Links. In *Design Automation Test in Europe Conference Exhibition (DATE)*, March 2010. 96

[177] Aoxiang Tang, Yang Yang, Chun-Yi Lee, and Niraj K. Jha. McPAT-PVT: Delay and Power Modeling Framework for FinFET Processor Architectures Under PVT Variations. *Transactions on Very Large Scale Integration (VLSI) Systems*, September 2015. 97, 98, 103, 104

[178] Radu Teodorescu and Josep Torrellas. Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors. In *International Symposium on Computer Architecture (ISCA)*. IEEE, June 2008. 98, 99

[179] Smruti R. Sarangi, Brian Greskamp, Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects. *Transactions on Semiconductor Manufacturing*, 21:3–13, February 2008. 98, 99

[180] Howard Chen, Scott Neely, Jinjun Xiong, Vladimir Zolotov, and Chandu Visweswariah. Statistical Power Analysis for High-Performance Processors. *Journal of Low Power Electronics (JOLPE)*, 5(1):70–76, April 2009. 98, 99

[181] Yaping Zhan, Andrzej J. Strojwas, Xin Li, Lawrence T. Pileggi, David Newmark, and Mahesh Sharma. Correlation-Aware Statistical Timing Analysis with Non-Gaussian Delay Distributions. In *Design Automation Conference (DAC)*. ACM, June 2005. 98, 99

[182] Magneti Marelli. System Validation. https://www.magnetimarelli.com/business_areas/powertrain/competences/system-validation, 2018. 109

[183] Michael S. Floyd, Phillip J. Restle, Michael A. SPerling, Pawel Owczarczyk, Eric J. Fluhr, Joshua Friedrich, Paul Muench, Timothy Diemoz, Pierce Chuang, and Christos Vezyrtzis. Adaptive Clocking in the POWER9 Processor for Voltage Droop Protection. In *International Solid-State Circuits Conference (ISSCC)*. IEEE, February 2017. 110, 111

[184] Chikafumi Takahashi, Shinichi Shibahara, Kazuki Fukuoka, Jun Matsushima, Yuko Kitaji, Yasuhisa Shimazaki, Hirotaka Hara, and Takahiro Irita. A 16nm FinFET Heterogeneous Nona-Core SoC Complying with ISO26262 ASIL-B: Achieving 10^{-7} Random Hardware Failures per Hour Reliability. In *International Solid-State Circuits Conference (ISSCC)*. IEEE, February 2016. 110, 111

[185] Keith A. Bowman, Carlos Tokunaga, Tanay Karnik, Vivek K. De, and James W. Tschanz. A 22 nm all-digital dynamically adaptive clock distribution for supply voltage droop tolerance. *Journal of Solid-State Circuits (JSSC)*, 48(4):907–916, April 2013. 110, 111

[186] Leonidas Kosmidis, Eduardo Quiñones, Jaume Abella, Tullio Vardanega, Carles Hernández, Andrea Gianarro, Ian Broster, and Francisco J. Cazorla. Fitting Processor Architectures for Measurement-Based Probabilistic Timing Analysis. *Microprocessors and Microsystems (MICPRO)*, 47:287–302, November 2016. 110, 113, 114, 115

[187] Ajay M. Joshi, Lieven Eeckhout, Lizy K. John, and Ciji Isen. Automated microprocessor stressmark generation. In *International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, February 2008. 111

[188] Sam Likun Xi, Hans Jacobson, Pradip Bose, Gu-Yeon Wei, and David Brooks. Quantifying Sources of Error in McPAT and Potential Impacts on Architectural Studies. In *International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, February 2015. 119

[189] Ayan Paul, Matt Amrein, Saket Gupta, Arvind Vinod, Abhishek Arun, Sachin Sapatnekar, and Chris H. Kim. Staggered Core Activation: A Circuit/Architectural Approach for Mitigating Resonant Supply Noise Issues in Multi-core Multi-power Domain Processors. In *Custom Integrated Circuits Conference (CCIC)*. IEEE, September 2012. 119

[190] Jianping Xu, Peter Hazucha, Mingwei Huang, Paolo Aseron, Fabrice Paillet, Gerhard Schrom, James Tschanz, Cangsang Zhao, Vivek K. De, Tanay Karnik, and Greg Taylor. On-Die Supply Resonance Suppression Using Band-Limited Active Damping. In *International Solid-State Circuits Conference (ISSCC)*. IEEE, February 2007. 119

[191] Antoon Purnal, Lukas Giner, Daniel Gruss, and Ingrid Verbauwhede. Systematic Analysis of Randomization-based Protected Cache Architectures. In *Symposium on Security and Privacy (S&P)*, May 2021. 125