DOCTORAL THESIS

Approaches to Extra-Functional Verification of Security and Reliability Aspects in Hardware Designs

Xinhui Lai

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2022

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
29/2022

Approaches to Extra-Functional Verification of Security and Reliability Aspects in Hardware Designs

XINHUI LAI

PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems

The dissertation was accepted for the defence of the degree of Doctor of Philosophy in Computer and Systems Engineering on 20 May 2022

Defence of the thesis: 17 June 2022, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Xinhui Lai

signature

European Union
European Regional
Development Fund

Investing
in your future

Copyright: Xinhui Lai, 2022
ISSN 2585–6898 (publication)
ISBN 978-9949-83-851-6 (publication)
ISSN 2585–6901 (PDF)
ISBN 978-9949-83-852-3 (PDF)
Printed by Koopia Niini & Rauam

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
29/2022

Riistvaraprojektide turva- ja töökindlusaspektide ekstrafunktsionaalse verifitseerimise lähenemisviisid

XINHUI LAI

TALTECH
KIRJASTUS

Contents

List of Publications

The present PhD. thesis is based on the following publications that are referred to in the text by Roman numbers.

I M. Jenihhin, X. Lai, T. Ghasempouri, and J. Raik, "Towards multidimensional verification: Where functional meets non-functional," in *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–7, IEEE, 2018

II X.Lai, A.Balakrishnan, T.Lange, M.Jenihhin, T.Ghasempouri, J.Raik, and D.Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," Microprocessors and Microsystems, vol.71, 2019

III X.Lai, M.Jenihhin, J.Raik, and K.Paul, "PASCAL: Timing SCA resistant design and verification flow," in 2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 239–242, IEEE, 2019

IV X.Lai, M.Jenihhin, G.Selimis, S.Goossens, R.Maes, and K.Paul, "Early RTL analysis for SCA vulnerability in fuzzy extractors of memory-based PUF enabled devices," in 2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC), pp.16–21, IEEE, 2020

V X.Lai, T.Lange, A.Balakrishnan, D.Alexandrescu, and M.Jenihhin, "On antagonism between side-channel security and soft-error reliability in BNN inference engines," in 2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1–6, 2021

Author's Contributions to the Publications

I In publication I, the author participated in the research problem formulation, performed the literature review, and carried out the case study. The author also contributed to the manuscript writing and presented the research work at the conference.

II Publication II is an extension of publication I. The author participated in the development of the idea involving collaboration with researchers from a partner company, and was in charge of the whole paper integration, revision and submission.

III In publication III, the author took part in the idea generation and was responsible for the development of the research idea. The author analyzed the potential side-channels in the design, developed the code for assertions and scripts to run the formal tool JasperGold Security Path Verification. Then the author developed the final experimental environment and conducted the experiment. The author wrote and finalized the manuscript and presented the research work at the conference.

IV In publication IV, the author participated in the research idea generation and established a new cross-sectoral collaboration. The author elaborated and developed the research idea and then was responsible for hardware design modification and proceeded with the experiment. Finally, the author wrote, finalized the manuscript, and presented the research work at the conference.

V In publication V, the author organized the collaborative research discussions with the partner company, was involved in the research idea generation and development, and carried out the analysis of the state of the art. The author analyzed the power side-channel attack mitigation techniques, applied the mitigation techniques to the case-study Binarized Neural Network, and contributed to the fault injection experiment. The author structured the manuscript, contributed to writing and finalized the manuscript, and presented the research work at the conference.

Abbreviations

AADL	Architecture Analysis and Design Language
AES	Advanced Encryption Standard
AI	Artificial Intelligence
ASIL	Automotive Safety Integrity Level
BCH	Bose–Chaudhuri–Hocquenghem
BMA	Berlekamp Massey Algorithm
BNN	Binarized Neural Network
BTI	Bias Temperature Instability
CART	Classification and Regression Trees
CMOS	Complementary Metal-Oxide Semiconductor
CPF	Common Power Format
CPS	Cyber-Physical System
CTL	Computation Tree Logic
DAQ	Data Acquisition
DNN	Deep Neural Network
DPA	Differential Power Analysis
DUV	Design Under Verification
EA	Euclidean Algorithm
ECC	Error Correction Code
EDA	Electronic Design Automation
ERT	Energy-aware Real-Time
FE	Fuzzy Extractor
FDR	Functional De-Rating
FEM	Finite Element Method
FIT	Failures In Time
FME(C)A	Failure Mode and Effects (Criticality) Analysis
FPGA	Field-Programmable Gate Array
HLS	High-Level Synthesis
HW	Hardware
IC	Integrate Circuit
IFT	Information Flow Tracking
IoT	Internet-of-Things
IP	Intellectual Property
JG SPV	JasperGold Security Path Verification
K-NN	K-Nearest Neighbors
LFM	Latent Fault Metric
LDR	Logical De-Rating
LTL	Linear Temporal Logic
LTR	Life-Time Reliability
LUT	Look-Up Table
ML	Machine Learning
MPSoCs	Multi-Processor System on Chips
MSB	Most Significant Bit
NoC	Network on Chip
NVM	Non-Volatile Memory
PASCAL	PAth based Side Channel AnaLysis
PSL	Property Specification Language

PUF	Physical Unclonable Function
RAS	Reliability, Availability and Serviceability
RBF	Radial Basis Function
RCA	Ripple-Carry Adder
RIIF	Reliability Information Interchange Format
RS	Reed-Solomon
RSA	Rivest–Shamir–Adleman
RSN	Reconfigurable Scan Network
RTL	Register Transfer Level
SAT	Satisfiability
SCA	Side-Channel Attack
SEE	Single-Event Effect
SER	Soft-Error Reliability
SET	Single-Event Transient
SEU	Single-Event Upset
SNR	Signal-to-Noise Ratio
SOTA	State-of-the-Art
SPFM	Single-Point Failure Metric
SRAM	Static Random Access Memory
SVA	System Verilog Assertion
SVR	Support Vector Regression
TDR	Temporal De-Rating
TLM	Transaction-Level Modeling
UPF	Unified Power Format
UVM	Universal Verification Methodology

1 Introduction

The next generation of computing, targeting an order of magnitude in performance improvement, is driving the development of electronic systems based on emerging nanoelectronic devices [2]. Benefiting from the evermore shrinking transistor technology, increasingly sophisticated electronic systems can be built on small-size chips, yet contain powerful functionalities. While this trend makes electronic systems suitable to many applications, it also increases their sensitivity to the environment, making these systems error-prone.

Electronic systems are used in applications ranging from commodity smartphones to advanced and expensive earth-orbiting satellites. These sophisticated and volatile scenarios are driving the evolution of the design of nanoelectronic systems. The latter is based on requirements for complex functional operations while guaranteeing non-functional (or extra-functional) requirements such as power-efficiency, reliability, and high levels of security. Unlike functional requirements, extra-functional requirements describe important constraints on the behavior of the system and specify certain qualities of the design [3].

For advanced nanoelectronic systems, maintaining specific extra-functional requirements is fundamental to successful implementation. For example, Internet-of-Things, which is featured by collecting and handling massive data from different devices through networks, implies exchanges of sensitive information and has created significant security concerns [4]. Here, ensuring the confidentiality and integrity of the data becomes the system's priority. On the other hand, for electronic devices used in high-radiation environments, such as in outer space, or for soft-error sensitive devices, such as memories and registers, relevant reliability considerations are brought to the front. As chip density increases, the rate of errors caused by cosmic rays increases significantly. The functional failures caused by soft errors are not negligible anymore, and soft-error reliability has become a limiting factor in design [5]. The prior facts have brought up the demand for novel design strategies used to improve the design implementation's quality (power, security, reliability, etc.).

The mentioned new requirements for design implementation have forced an evolution in design verification methodology. As indicated in [6], fulfilling designs' extra-functional requirements is critical to the success of the target product. However, there is no consensus in the community about their definition, and how one should elicit, document and validate the extra-functional requirements. Unlike functional verification, which has an established methodology, extra-functional verification is a relatively new challenge. Since the contained extra-functional aspects are abstract, defining the related behavior from the design implementation is obscure and not sufficiently supported by the established state-of-the-art tools and methodologies.

This thesis provides a study of the verification of extra-functional aspects with an emphasis on critical aspects such as security and reliability.

1.1 Motivation

Modern electronic technology has propelled the extensive usability and high complexity of electronic devices. Over the years, the functionality of electronic design has always been the main concern for engineers [7]. However, with safety-critical applications and cyber-physical systems being widely adopted, extra-functional aspects of security, reliability, and power have been frequently addressed in the research. Thus, verification of extra-functional aspects is getting to the front.

For a specific extra-functional aspect, the related behavior is normally composed of measurements of the sequences of functional behavior and there is no straightforward way to obtain complete and correct functional sequences. Therefore, manual or supervised analysis of the targeted extra-functional aspect and extraction of related functional sequences has become a challenge for the verification process. In addition, the complexity of the involved functional behaviors increases alongside the design abstraction levels and design complexity, thus introducing compounded difficulties. Furthermore, tools and methods applicable for extra-functional verification are limited [8]. Due to the increased complexity of the current designs, the corresponding extra-functional behaviors tend to be sophisticated. It is even more difficult to extract the extra-functional behavior through existing methods or tools. Although the extra-functional behavior is extracted from the functional behavior, the established metrics used in functional verification can not be applied in extra-functional verification.

Facing the new paradigm for design verification, there is a lack of systematic study of the extra-functional aspects and the related design verification techniques while these are well-defined in software disciplines. Unfortunately, the definition and classification for extra-functional aspects in software can not be directly reused for hardware design due to the rather distinctive design description concepts. Therefore, there is a need to understand the extra-functional aspects from the hardware perspective and explore novel hardware verification methodologies to address different extra-functional aspects.

Hardware security is one of the most critical extra-functional aspects. It is featured to protect the secret information stored in physical devices or the hardware designs' information from being stolen and corrupted during different design stages. It has attracted notable attention in the advanced electronic systems discipline [9,10]. However, vulnerabilities causing security problems that originate from design flaws are rarely considered during design space exploration. A novel type of attack, named side-channel attack, has proven efficient for inferring sensitive information by exploiting unintended information leakage of the crypto core implementation. Most of the works for side-channel attack verification mentioned in the literature address side-channel vulnerability only after manufacturing, since the exploitable side-channel information (timing, power, and electromagnetic) is acquired by measuring the real devices. Such methods imply a huge expense for re-design and re-production for the electronic systems. Therefore, identifying the side-channel vulnerability at the early design stage would be of great value for the time-to-market production as well as for reducing design costs.

Along with the requirements for security, today's critical systems are required to be capable of providing correct results even in the presence of hardware-level faults. Therefore, during the design stages, engineering tends to apply additional security and reliability mechanisms to the design separately. However, for the whole system, there is no guarantee that the introduced security mechanism does not influence the system's reliability and vice versa. It is hard for the designer to identify the intersection between applied security and reliability at the design phase. Overall, there is still a lack of methodologies for multiple extra-functional aspects verification.

1.2 Research Objectives

Considering the above problems, the **research objectives** of this PhD thesis are as follows:

- Providing a systematic study of extra-functional aspects in hardware design and understanding the challenges in this research area.

- Proposing novel methodologies to support verification of the security and reliability aspects at early design phases. The goal is to reduce the time to market and cost for nanoelectronic systems manufacturing.
- Exploring the mutual influence of extra functional aspects in practice, considering the security and reliability aspects as a case study.

1.3 Contributions

Targeting the above research objectives, the contributions of this PhD thesis are summarized as follows.

- Addressing the systematic study for extra-functional aspects verification, this PhD thesis has **proposed the first systematic review on** extra-functional verification for hardware designs. The study is based on a review of the existent research works and has the following contributions:
  - proposes an up-to-date taxonomy for extra-functional aspects' classification from the hardware perspective;
  - presents a state-of-the-art survey addressing different extra-functional aspects and summarizing the detailed information about the related verification methods;
  - identifies the research challenges for today's extra-functional aspects' verification and proposes an initial approach towards multiple extra-functional aspects verification relying on machine learning techniques.
- Targeting the security aspect, this thesis focuses on timing side-channel attacks and brings identification of relevant design vulnerability up to the early RTL design level instead of the physical design level addressed in the literature. In particular, this work:
  - provides an early RTL analysis to identify the timing side-channel vulnerability, which is normally addressed after manufacturing in the literature;
  - presents both simulation-based and formal verification methods to analyze the vulnerability;
  - indicates the limitation of the two verification methods regarding early-stage analysis of timing side-channel attack;
  - provides a lightweight and effective mitigation technique for the timing side-channel vulnerability identified for a Rivest–Shamir–Adleman (RSA) crypto core implementation.
- Focusing on the reliability, the thesis is the **first time** (to the best of our knowledge) to address the mutual influence between security and reliability of the Binarized Neural Network (BNN) network applied to a critical system. This work:
  - presents an analysis for the soft-error reliability jeopardy by the side-channel related security mitigation;
  - provides in-depth reasoning about how the security mechanism influences the entire system's reliability;
  - emphasizes the interdependency of the design's reliability and security aspects.

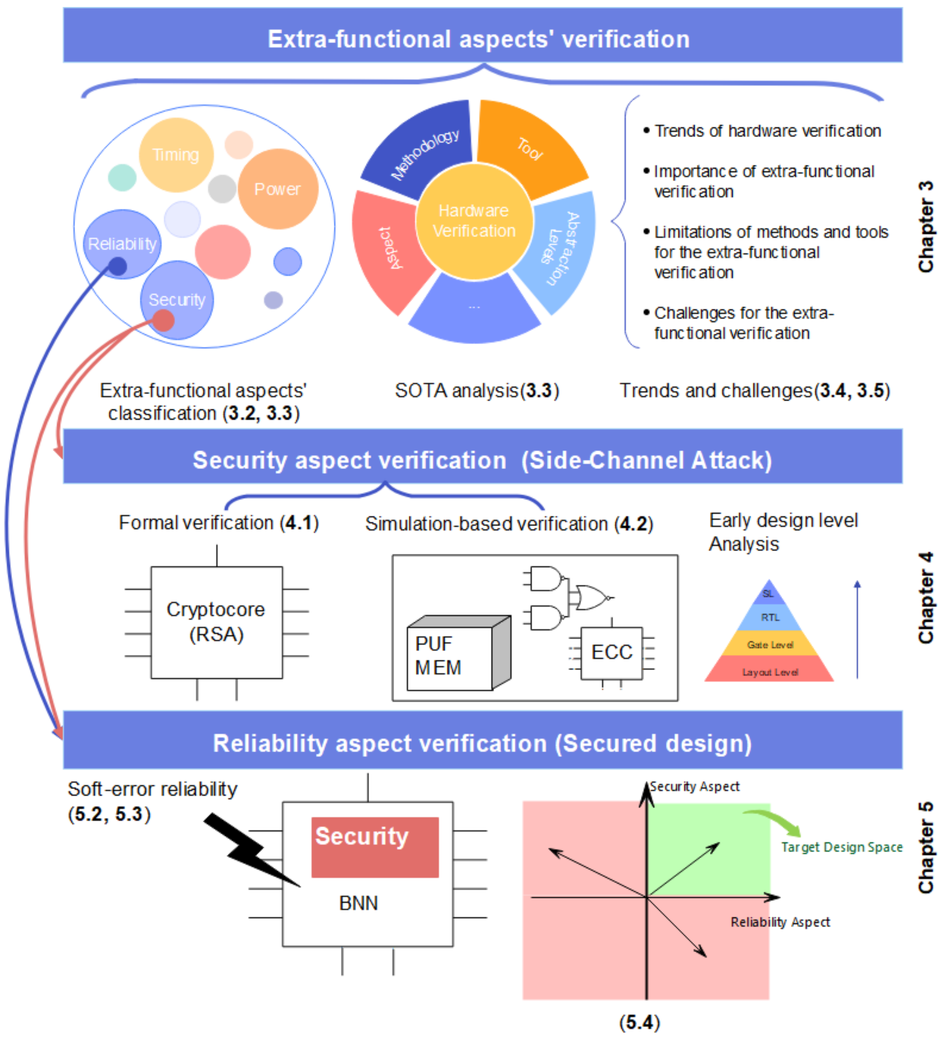

Extra-functional aspects' verification
Timing
Power
Reliability
Security
Methodology
Tool
Hardware Verification
Abstraction Levels
Aspect
...
• Trends of hardware verification
• Importance of extra-functional verification
• Limitations of methods and tools for the extra-functional verification
• Challenges for the extra-functional verification
Chapter 3
Extra-functional aspects' classification (3.2, 3.3)
SOTA analysis(3.3)
Trends and challenges(3.4, 3.5)
Security aspect verification (Side-Channel Attack)
Formal verification (4.1)
Simulation-based verification (4.2)
Early design level Analysis
Cryptocore (RSA)
PUF MEM
ECC
SL
RTL
Gate Level
Layout Level
Chapter 4
Reliability aspect verification (Secured design)
Soft-error reliability (5.2, 5.3)
Security
BNN
Security Aspect
Target Design Space
Reliability Aspect
(5.4)
Chapter 5

Figure 1: Thesis Contributions and Organisation

1.4 Thesis Organization

This thesis has six chapters. The first chapter presents the introduction of the thesis. Chapter 2 discusses the background. In this chapter, the basic knowledge of hardware verification, functional and extra-functional requirements and methods used for verification are introduced. The main contributions of the thesis are depicted in Figure 1. As shown in the figure, Chapter 3 targets the study of extra-functional aspects' verification. This chapter provides a study of the state of the art for extra-functional aspects and proposes a taxonomy of the extra-functional aspects from the hardware point of view. Then it identifies the trends and challenges for the methodology used for extra-functional aspects' verification. In order to have an in-depth understanding of the methodology used for the extra-functional aspects, this thesis focuses on security and reliability aspects for further study, which are presented at Chapter 4 and Chapter 5 separately. For the study of security verification, Chapter 4, it focuses on the side-channel attack and provides an early RTL analysis. It includes the timing analysis of a crypto core and the subblock of the system containing a security mechanism, i.e., an error-correction decoder used in a memory-based Physical Unclonable Function (PUF) scenario. In this work, both formal and simulation-based verification approaches are provided. Chapter 5 studies the reliability aspect verification of a hardware BNN implementation that is resistant to power side-channel attacks. As shown at the bottom of Figure 1, it explores the impact of a security-enhanced design on its reliability. The main purpose of this study is to explore the design space considering both reliability and security. Chapter 6 summarizes and concludes this PhD thesis. Additionally, the possible future work towards a holistic verification method for multiple extra-functional aspects is outlined.

2 Background

2.1 Hardware Verification

The design process of digital Integrated Circuit (IC) is usually refined through the top-down design abstraction levels to deal with the design complexities. The traditional design flow is shown in Figure 2. In the design flow, the verification process takes the role of checking whether a given design is correctly implemented according to the specification. And it has the highest impact on the essential business drivers, i.e., quality, schedule and cost. However, the verification process is very time-consuming. As the statistical data shows, 70% of the development cycle is used for the verification step [11].

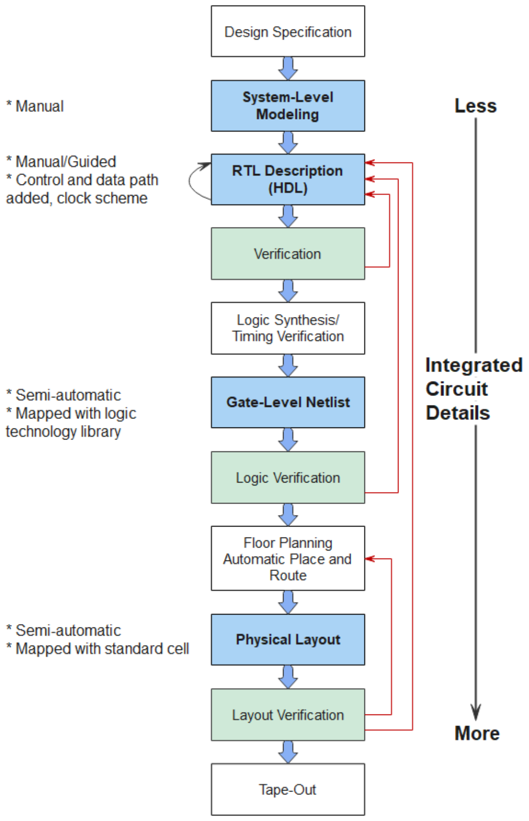


Figure 2: Digital IC Design Flow

As illustrated in Figure 2, the general design flow can be divided into four stages, which are marked in the blue boxes: System Level, Register Transfer Level, Gate Level and Physical Layout Level. For each design stage, the verification process, marked in a light green color at the same design level, is performed after the design step and aims to ensure the implementation is designed as specified. If any design bugs are found during verification, it will result in modification or redesigning of the implementation at the current abstraction level or even the previous design levels. After modification, the corrected design needs to repeat the verification procedure all over again. Therefore, it is critical to ensure that all the design bugs are caught at each stage completely.

Along the abstraction spectrum of the design, more design details are added throughout the implementation process, which makes the design behavior closer to the real circuit but also more complex. Methods and tools also vary at different design stages. Figure 3 indicates the design objectives and design verification at different abstraction levels.

- **System Level**: The design objective for the system level is design partitions. It focuses on the functional implementation of different blocks as well as the interconnect infrastructure of the system. Thus, for the system-level verification, besides the functional verification for each individual IP blocks, it also needs to verify the interconnection between blocks. However, some of the IP blocks and interconnection infrastructures are reusable. Thus the verification solution for these blocks can be reused.

- **RTL**: At RTL, microarchitecture blocks implemented by VHDL/Verilog/System Verilog are filled into the design to provide hardware details. Hardware verification starts from this stage. There are well-defined verification strategies for functional verification such as coverage-driven verification targeting structural coverage metrics (statements, branches, conditions, etc.) and functional coverage modelled by asserted functional properties.

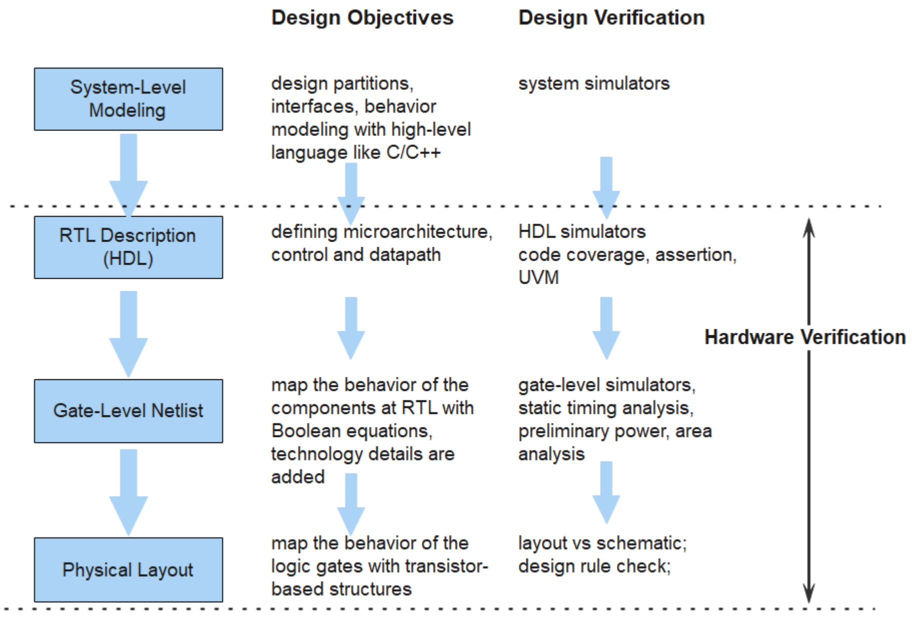

Design Objectives
Design Verification
System-Level Modeling
design partitions, interfaces, behavior modeling with high-level language like C/C++
system simulators
RTL Description (HDL)
defining microarchitecture, control and datapath
HDL simulators code coverage, assertion, UVM
Hardware Verification
Gate-Level Netlist
map the behavior of the components at RTL with Boolean equations, technology details are added
gate-level simulators, static timing analysis, preliminary power, area analysis
Physical Layout
map the behavior of the logic gates with transistor-based structures
layout vs schematic; design rule check;

Figure 3: Design and Verification for Different Abstraction Levels

- **Gate Level**: At this stage, the technology library is used to map the functional behavior to the Boolean equation. It contains the static timing, area, and power consumption details of each logic gate. Therefore, besides the functional verification, it is also possible to provide preliminary power and area analysis and static timing analyses.

- **Physical Layout Level**: The last design stage is at the Physical Layout Level. At this stage, the behavior of logic gates is translated to a transistor-based structure. At this level, circuit details like floorplan, place and route, etc. become available.

Design complexity increases with descending the design abstraction levels, and the verification time is increasing as well since the tools and methods used for design verification need to go through additional design details. Therefore, it is cheaper to eliminate design bugs related to functional and extra-functional aspects at RTL and Gate-Level while the cost of verification grows at the lower levels, such as the physical layout level. Thus, it is preferred to find design bugs at the earlier design stage. In this thesis, the research work focuses on hardware design verification at RT and Gate levels.

2.2 Functional vs Extra-functional

2.2.1 Functional and Extra-functional Requirements

The design specification defines both the functional and extra-functional requirements. **Functional requirements** describe the targeted functional tasks that should be implemented, while the **extra-functional requirements** describe the important constraints upon the functional behavior of the system and specify a range of design qualities [3].

Rapid technology development has fuelled ever-increasing complexity in functional as well as extra-functional requirements of electronics, which in return enables its usability to diverse applications and hostile environments. The requirements for modern nanoelectronics system designs tend to be complex and rigorous. The systems are required to realize correct functions while maintaining the qualities defined by the extra-functional requirements. In the software discipline, extra-functional aspects affect the overall architecture of the system rather than the individual components and they can be categorized into product requirements, organizational requirements and external requirements [12]. In hardware design, each extra-functional aspect strongly depends on the design type, targeted applications and users' requirements. Following the design paradigm shifts, several extra-functional aspects have received significant academic research attention and are loosely summarized as follows.

- **Security** [13]: Hardware security lies essential in ensuring trust, integrity and authenticity of the integrated circuit and electronic systems [13]. The related research fields combine multiple knowledge domains, which include discrete math, algorithm design and transformation, digital architecture design with analog twists and controlled production technologies [9]. Based on different hardware implementations, the related security issues are 1) implementation-independent vulnerability in cryptographic chips, which can lead to information leakage; 2) distributed supply chain for manufacturing and distribution of electronic components, which might lead to malicious modification of IC in an untrusted design house or foundry and also referred as Hardware Trojan attacks; 3) side-channel attacks, aimed at secret information extraction through measurement and analysis of side-channel information such as power, signal propagation delay (timing) and electromagnetic emission; 4) security issues related to hardware designs that are used to provide an appropriate level of isolation between secure and insecure data in order to protect sensitive assets stored in hardware from malicious software and network.

- **Reliability**: The reliability defines the ability of the design to function correctly in the field over a certain period of time under predefined conditions [14]. The key drivers for the reliability aspect in today's designs are recent industrial standards

(IEC61508, ISO26262, etc.) in different applications like the self-driving car, space-based electronic system, etc. The dominant reliability issues for a nanoscale integrated circuit are 1) the aging issue, e.g., by Bias Temperature Instability, that lead to delays and, ultimately, to permanent faults; 2) soft errors related single event upset or single event transient faults when ionized particles from cosmic rays or terrestrial sources strike on sensitive regions of the semiconductor devices. These faults may hamper the proper functionalities of the circuit [15,16]. However, the vulnerability of each IC design to the reliability issues can be verified already at the design phase.

There are also other extra-functional aspects like power, timing, etc. that are discussed in detail in **Chapter 3**.

2.2.2 Functional and Extra-functional Aspects' Verification

Mapping to the design phase, the corresponding functional and extra-functional requirements should be verified before manufacturing. The objective of functional verification is to check whether the hardware design conforms to the functional specification, and it is fundamental to successful design implementation. Typically, verification engineers work on functional verification predominantly at the RTL of the design rather than lower levels like Gate Level or Physical Layout Level. At this stage, functional requirements are either translated into test input/output stimuli pairs or properties for further verification. Methods used for functional verification can be formal or simulation-based approaches, and the details are indicated in Section 2.3. But either the stimuli or the properties are explicit and easy to be extracted from the specifications. For the simulation-based method, there are also well-established metrics to quantify the verification completeness. However, for extra-functional aspects, the related design verification methods are still under development.

There is a rising number of research works exploring the identification and verification methods for extra-functional aspects. Unlike explicit functional behavior, extra-functional behavior is implicit and relies on sequences of correct functional behavior. As the side-channel attack of security aspect, the side-channel information (timing, power, or electromagnetism) extracted from the design is based on the measurements of the correct functional sequences of the implementation during the normal workloads. For different functional aspects, there is no uniform guidance to define their behaviors, which means all of them need to be studied individually. However, for some extra-functional aspects like reliability, the methodology and evaluation are well-defined by the fault injection method.

2.3 Verification Methodology

All verification methods can fall into the following two categories: formal and simulation-based verification. The main difference between formal and simulation-based verification is the rigorous mathematical proof. The simulation-based verification method relies on the defined input stimuli and therefore, it can not guarantee verification completeness. However, the formal method is supported by a mathematical model and it is proven complete.

2.3.1 Simulation-Based Verification

Simulation-based verification is the most commonly used verification approach and is widely used for functional verification. A general flow of simulation-based verification is illustrated in Figure 4. Starting from a hardware design, there are three procedures running in parallel [11]:

- **Linting**: Before simulation, the design needs to go through the linter program, which basically runs programmatic and stylistic error check for the source code and help remove coding style violations. It is a static check and does not need any input stimulus.
- **Test bench design**: The test bench design is used to mimic the environment, in which the design will reside, and it will not be assembled to the final design. As for functional verification at RTL, the test bench is written in HDL, and its purpose is to apply input waveforms to the design and monitor the output changes under a defined working environment during the simulation and verification step. Normally, it has by far fewer coding style constraints and might include pre-analysis of the results.
- **Test Plan**: The test plan helps with the input vector stimulus generation. The input vectors targeted at specific functionalities and features are called *directed tests*. The ultimate goal for the test plan is to generate the test vectors, which can cover the entire input spaces, evoke all the functionalities and then expose the misbehavior of the design. It is applicable to small-size designs but infeasible for a larger design since the input design space grows exponentially with the size of design input.

The linting process purely depends on the compiler and normally is embedded in the simulation tools like Modelsim and Questasim. But the test bench design and test plan are designed and written by the designer based on specification and knowledge of verification methodology.

The simulation quality of a design can be measured by the coverage metrics. The coverage measures how much the design is simulated and verified. As mentioned previously, it is impractical to simulate all tests, and the coverage metrics like code coverage, statement coverage, functional coverage, etc., which are used to quantify the influences to the designs by applying the test stimulus. The idea of coverage metrics is to provide certain level of confidentiality of the executed verification for design under verification, provide feedback for the current input vectors and then guide the engineers to generate better stimuli.

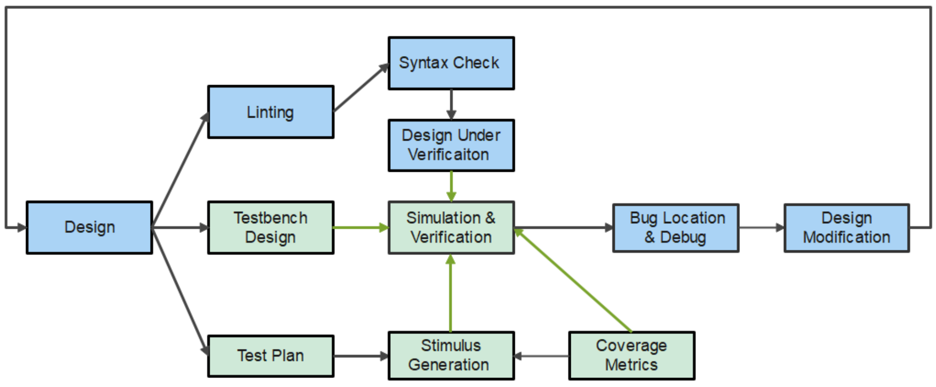


Figure 4: Simulation-Based Verification Flow

2.3.2 Formal Verification

Unlike simulation-based verification, formal verification does not require test vectors and is based on mathematical analysis, which ensures complete behavior exploration. Thus it is in favor of completeness of design space exploration, which generally can not be covered by simulation-based verification. However, practically, formal verification is only suitable for moderate-size designs like blocks or modules rather than complex electronic systems. Because the software used for formal verification commonly costs extensive memory and needs long runtime sometimes before making a final verification decision [11].

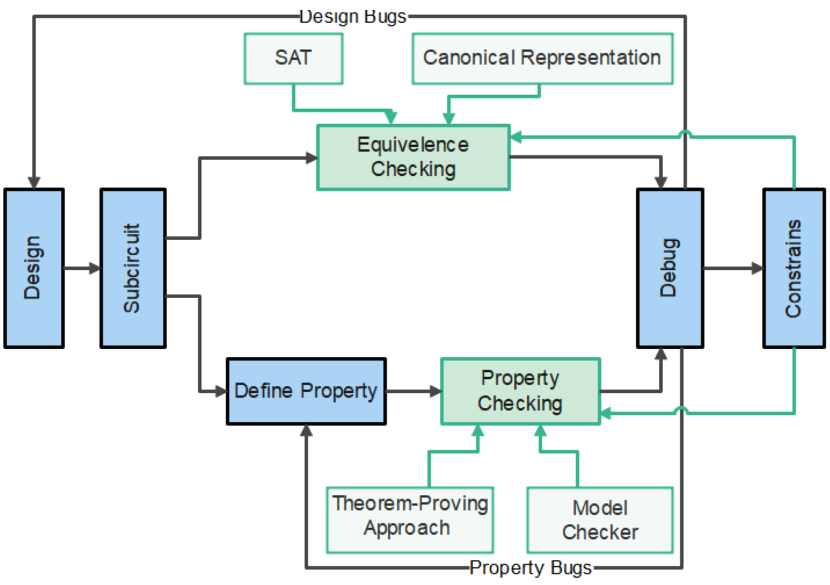


Figure 5: Formal Verification Flow

The general formal verification flow is shown in Figure 5. The verification process starts with the design specification. Formal methods can be classified into property checking and equivalence checking.

- Equivalence Checking is the process of determining whether the two implementations are functionally equivalent. Two approaches are widely used here. The first is SAT (satisfiability). It searches the input space systematically in order to find at least one input vector that could distinguish two implementations. The other one distinguishes the two designs by comparing their canonical representations since the canonical representation has the characteristic property that the two logical functions are equivalent if and only if their respective representations are isomorphic.

- Property Checking takes properties extracted from design specification to prove or disprove the correctness of the regarding properties. Normally, the checking procedure is done inside a model checker, which refers the related design as a computational model. The principle of property checking is to search the entire state space and try to address the failure point of the property. If such a point is found, the property is proven to be failed, and a countermeasure will be provided. Otherwise, the property is satisfied. In practice, the model checker can have another time-out situation, which means the property could not be defined within the defined checking time. Usually, these cases can happen when the selected design is too complex for the tool, or the defined property does not hold.

As mentioned, the formal method is efficient for small-sized designs. When applying equivalence checking, the input space for the block design is not limited, and it should have constraints regarding the former connected circuit. Thus, related constrains should be added manually. Property checking is akin to equivalence checking. The process is not automatic, and the engineers need to understand the tool and define the proper constrains for the property.

Equivalence checking and property checking can be both used for functional verification. However, as for extra-functional aspect verification, property checking still can be easily applied. E.g., in this thesis, the research work shown in Chapter 4 applies formal verification.

3 Multidimensional Verification

In this chapter, state of the art of multidimensional verification is studied. Based on the analysis, a taxonomy of multidimensional hardware verification aspects and a survey classifying the research works is proposed. Additionally, trends in different extra-functional aspects of verification are summarized to show the verification methods' diversity. Finally, a machine learning based method for possible multidimensional aspects' verification is proposed and demonstrated on a case study.

This chapter is based on the following publications:

- **I** M. Jenihhin, X. Lai, T. Ghasempouri, and J. Raik, "Towards multidimensional verification: Where functional meets non-functional," in *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–7, IEEE, 2018

- **II** X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," in Microprocessors and Microsystems, vol. 71, 2019.

3.1 Introduction

Several recent prominent trends in electronic systems design can be observed. Safety-critical applications in the automotive domain set stringent requirements for electronics certification, the Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) devices are immersed in physical environments, significantly constrained in resources and expected to provide levels of security and privacy [17], ultra-low power feature or high performance. Very complex electronic systems, including those built from the non-certified for reliability commercial-off-the-shelf components, are used for safety- and business-critical applications. These trends along with gigascale integration at nanoscale technology nodes and multi-/many-processor based systems-on-chip architectures have ultimately brought to the front various extra-functional aspects like security, reliability, timing, power consumption, etc., of the electronic systems' design at the chip design level.

Accordingly, for proper design model verification, it is necessary to detect design errors not only affecting functional but also extra-functional aspects of the target electronic system. Such aspects are also interchangeably referred to as non-functional. Theoretically, the sole task of extra-functional verification of a design model is to detect deviations that could cause violations of extra-functional requirements. However, in practice, it often intersects with the task of functional verification [18, 19] and is challenging to extract complete behaviors from the target design model.

This chapter is organized as follows. Section 3.2 presents a taxonomy of multidimensional verification aspects. Sections 3.3 proposes a state-of-the-art survey with the key trends in verification for the main extra-functional aspects. Section 3.4 discusses the multidimensional verification challenges and presents a motivational example for the functional and power verification dimensions. Section 3.5 proposes adoption of machine learning techniques for support of design's multi-aspect features extraction and verification. Finally, Section 3.6 draws the conclusions for the chapter.

3.2 Taxonomy of Multidimensional Verification Aspects

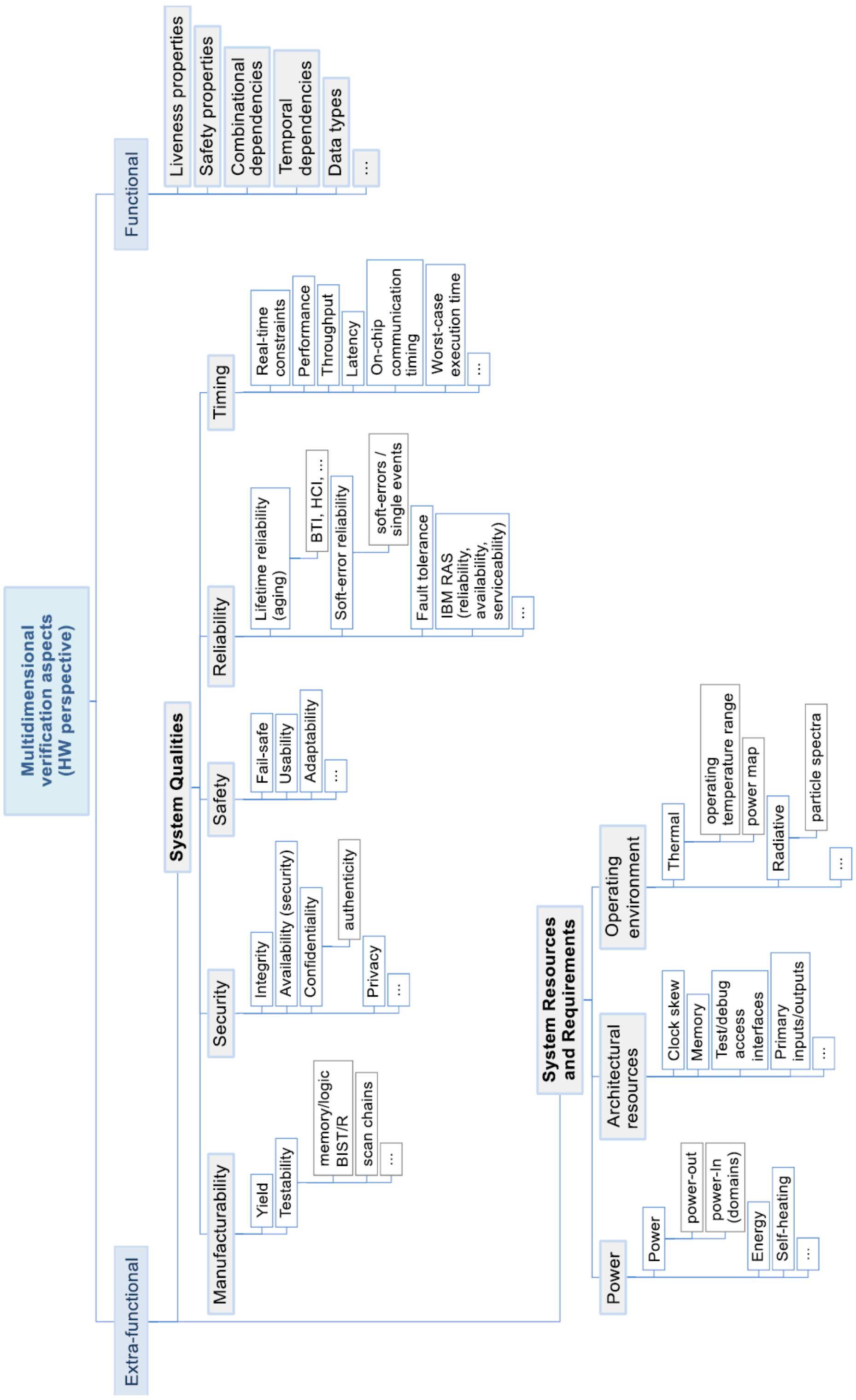


Figure 6: Taxonomy of Multidimensional Verification Aspects

In practice, relevance of each functional and extra-functional aspect strongly depends on the design type, target system application and specific user requirements. Following the design paradigm shifts, a number of extra-functional aspects have recently received significant academic research attention e.g., security. At the same time, there already exist established industrial practices for measuring and maintaining particular design qualities, e.g., the RAS (Reliability-Availability-Serviceability) aspect introduced by IBM [20]. In the software engineering discipline, the taxonomy of extra-functional requirements has a comprehensive coverage by the literature [6, 21–25]. However, it cannot be directly re-used for the HW verification discipline because of significant differences in the design models. Thus, we have proposed a taxonomy of multidimensional verification aspects in hardware perspective derived from the performed literature review and indicated in Fig 6. The conventional functional concerns are safety and liveness properties, combinational and temporal dependencies along with data types, however this list can be extended for particular designs. The extra-functional aspects can be strictly categorized into two groups: System Qualities and System Resources and Requirements (in bold). The main system qualities for extra-functional verification are manufacturability of the design, security, in-field safety, reliability during the operational lifespan and a set of timing aspects. The second group embraces the power and architectural resources as well as design constraints set by the operational environment.

Compared with software engineering discipline, several extra-functional aspects such as manufacturability, i.e., primarily yield and testability against manufacturing defects, fault-tolerance, reliability (subject to transient, intermittent and permanent hardware faults) and several aspects from the System Resources and Requirements group do not have a direct correspondence because of the distinct nature of faults and specification violations. Other aspects such as real-time constraints are very similar between the two domains.

3.3 Trends in Extra-functional Verification

Table 1 presents a survey of recent publications targeting extra-functional and multidimensional verification. Here, along with the specific extra-functional aspects details about the design model and verification approach are outlined, i.e., the design under verification type, verification engine, the level of abstraction, design representation language, compute model and the tool operated in the research. Then we focus on understanding trends for the extra-functional aspects that have the strongest attention in literature, i.e., security, in-field reliability, timing and power.

3.3.1 Security Aspects

Security is difficult to quantify as today there are no commonly agreed metrics for this purpose [17]. The key targeted security services [26] commonly represented as extra-functional aspects for verification are confidentiality, integrity and availability. Verifying security aspects is highly dependent on the type of attack and the attacker model assumed.

Many of the existing works in security verification (e.g., [27–31]) are focusing on the integrity attribute, mostly addressing hardware trojan detection. There also exist works that additionally target [32–36] or are exclusively considering [37, 38] the confidentiality aspect. Several solutions in security verification are restricted to specific target architectures or types of modules such as Reconfigurable Scan Networks (RSNs) [34, 37] or macro-asynchronous micro-synchronous pipelines [31]. To that end, for complex hardware architectures (e.g., large IEEE1687 Reconfigurable Scan Networks or MPSoCs) the specific on-chip security features to be verified also tend to be very sophisticated. These may include on-chip mechanisms for attack prevention (firewalls, user management, communications' isolation), attack protection (traffic scrambling, encryption) and attack resilience (checkers for side-channel attacks, covert channel detection, attack recovery mechanisms). Several works consider security verification for NoC-based MPSoCs. [39] proposes a method to formally verify the correctness and the security properties of a NoC router. Some solutions in the security verification of NoCs do indirectly address reliability due to the fact that they implement hardware monitors that allow avoiding both, attacks and in-field faults [27, 33].

According to recent surveys [40] and [41], cache access driven side-channel attacks have become a major concern in hardware security. In modern processors, deep hierarchy of cache memory is implemented to increase system performance. However, this makes modern computing systems, including IoT devices, vulnerable to cache side-channel attacks. There exist several works addressing verification of the cache security. In [42], the authors propose Computation Tree Logic (CTL) based modeling of timing-driven and access-driven cache attacks. This work concentrates on formally describing the attack types. Zhang and Lee [43] model cache as a state machine and propose a metric based on the non-interference condition to evaluate the access-based cache vulnerability. Canones et al. [44] propose a model to formally analyze the security of different cache replacement policies. None of the above-mentioned works consider multiple dimensions, or aspects.

An approach that is designed for modeling a multitude of extra-functional aspects is the model-based engineering example of Architecture Analysis and Design Language (AADL) [32]. While, in principle, AADL allows representing several extra-functional aspects (called quality attributes in AADL), Hansson et al. [32] only concentrate on analysis of confidentiality as a part of verifying security in a system with multiple levels of security. The authors in [45] have targeted a general Uppaal Timed Automata based multiview hardware modeling and verification approach taking into consideration of the security view.

Table 1: Survey of the SOTA Solutions for Extra-Functional and Multidimensional Verification

Pub.	Year	Extra-functional aspect[a]					DUV[e]	Ver. Eng.[f]	Abst. Level[g]	DRL[h]	Comp. Model[i]	Tool[k]
		S[j]	R[b]	T[c]	P[l]	O[d]						
[32]	2009	②①	-	-	-	-	HW/SW system	formal, correct-by-construction	SL	AADL	-	OSATE
[39]	2018	①	-	-	-	-	NoC	unbounded model-checking	RTL	VHDL/ Verilog, PSL	-	-
[33]	2016	②①	○	-	-	-	NoC	simulation, HW monitors	RTL	VHDL/ Verilog	-	-
[27]	2014	②	○	-	-	-	NoC	formal	GL	VHDL/ Verilog	-	SurfNoC
[34]	2017	①②	-	-	-	-	RSN	model check	RTL	ICL	Craig interpo-lation	CIP solver
[28]	2015	②	-	-	-	-	SoC	simulation	RTL	VHDL/ verilog	-	-
[35]	2016	②①	-	-	-	-	ALU	equivalence check	GL	-	OBF-SAT	-
[29]	2017	②	-	-	-	-	SoC	Semi.	GL	-	-	JasperGold SPV
[37]	2016	①	-	-	-	-	RSN	model check	RTL	ICL	Craig interpo-lation	CIP Solver
[38]	2017	①	-	-	-	-	control sys-tem	formal	SL	ASLan++	-	CL-AtSe

Continued on next page

Table1 – continued from previous page

Pub.	Year	Extra-functional aspect[a]					DUV[e]	Ver. Eng.[f]	Abst. Level[g]	DRL[h]	Comp. Model[i]	Tool[k]
		S[j]	R[b]	T[c]	P[l]	O[d]						
[30]	2017	②	-	-	-	-	IP cores	semi.	GL	VHDL	-	Mini-SAT
[31]	2015	②	-	-	-	-	ISA, Pipeline	model check	RTL	-	CTL, LTL	nuXmv SMV
[42]	2018	①	-	-	-	-	cache	model check	SL	-	CTL	-
[43]	2014	①	-	-	-	-	cache	model check	RTL/SL	-	FSM	Murphi
[44]	2017	①	-	-	-	-	cache	model check	RTL/SL	-	FSM	CacheAudit
[36, 46]	2013	②①	-	-	-	-	IPs and SoCs	formal	RTL, GL	Verilog	-	JasperGold SPV
[45]	2018	●	●	-	-	-	MPSoC	model check	SL, RTL	-	Timed Au-tomata	UPPAAL
[47]	2017	-	●	-	-	-	CPS	model check	SL	AADL	Timed Au-tomata	UPPAAL
[48]	2015	-	④	-	-	-	IP cores	formal	GL/RTL	LDDL	LDDL	Coq
[49]	2010	-	④	-	-	⑧⑨	processor	fault inject	GL	verilog	-	IBM in-house
[50]	2016	-	●	-	-	⑧⑨	SoC	fault inject	RTL	-	-	-
[51, 52]	2016	-	○(③)	-	○	⑬	smart system	simulation	SL	IP-XACT, systemC-AMS	-	-
[53]	2018	-	●	-	-	-	CPS	formal/ sim-ulation, HW monitors	RTL	VHDL	-	-
[54]	2014	-	④	-	-	-	IPs	SAT Solver	RTL	VHDL	-	-
[55]	2010	-	④	-	-	-	IPs, proces-sor	Simulation	RTL	VHDL/ Verilog	-	-
[56]	2014	-	④	-	-	-	memory	circuit-level simulation	circuit level	-	-	INFORMER
[57, 58]	2018	-	-	⑥	○	-	NoC	fault injection	RTL	VHDL	-	QoSinNoC
[59]	2011	-	-	⑤	-	-	memoory	model check	RTL	REAL/ AADL	-	Ocarina
[60]	2014	-	-	⑤	-	-	Scheduler of RT. system	model check	-	Promela	Time Petrinet	SPIN
[61]	2010	-	-	⑫	-	-	RT emb.system	model check	SL	AADL	-	YICES
[62]	2017	-	-	⑦	○	-	NoC, HW/SW architecture	simulation	SL	Graph As-sembly Lan-guage	Conne. Graphs	ArchOn
[63]	2012	-	-	-	●	-	IPs	simulation	SL	SystemC	-	-
[64]	2016	-	-	-	●	-	DSP cores	simulation	SL, GL, RTL	SystemC	-	Powersim
[65]	2017	-	-	⑤	○	-	automotive CPS	model check	SL	C, EAST-ADL	Timed Au-tomata	UPPAAL, SDV
[66]	2016	-	-	-	●	-	IPs	Semi., ABV	RTL	VHDL/ Verilog, SystemC	Hidden Markov Model	-
[67]	2012	-	-	⑪	○	-	distributed emb.system	simulation	SL	SystemC	-	-
[68]	2016	-	-	⑦⑤	○	⑬	HW/SW platform	semi.	RTL, TLM, SL	UML, C++, VHDL, SystemC-AMS	HIF	HIFSuite
[69]	2009	-	-	⑩	-	-	SoC/FPGA	simulation	RTL	Verilog/ VHDL	-	Modelsim
[70]	2018	-	-	⑩	-	-	NoC	simulation	RTL	System Verilog	-	UVM
[71]	2014	-	-	-	-	⑭	SoC	symbolic model check-ing	RTL, TLM	Verilog	-	Incisive Formal Verifier
[72]	2016	-	-	-	-	⑮	processor	simulation	ISA	ruby	-	McVerSi
[73]	2011	-	-	-	●	⑬	SoC	simulation	SL, GL, RTL	SystemC	-	Power-Mixer, -Depot, -Brick

Continued on next page

Table1 – continued from previous page

Pub.	Year	Extra-functional aspect^a					DUV^e	Ver. Eng.^{f}	Abst. Level^g	DRL^h	Comp. Model^i	Tool^k
		S^j	R^b	T^c	P^l	O^d						
[74]	2015	-	-	-	●	-	-	simulation	SL,TLM	SystemC	-	Power Kernel Tool
[75]	2011	-	-	-	●	-	SoC	simulation	SL	SystemC	-	Powersim

^{a} ● - this aspect is the main focus in the paper; ○ - this aspect is partially addressed.
^{b} R: Reliability (③ - lifetime reliability; ④ - soft-error reliability.)
^{c} T: Timing (⑤ - real-time constraints;⑥ - communication time constrain; ⑦ - performance.)
^{d}O: Other Aspects (⑧ - availability; ⑨ - serviceability; ⑬ - Thermal; ⑭ - Connectivity; ⑮ - Memory Consistency.)
^{e} DUV - Design Under Verification.
^{f} Ver. Eng - Verification Engine; Semi. - semi-formal.
^{g} Abst. Level - Abstract Level;GL - gate level; SL - system level; ISA - instruction set architecture level; TLM - transaction level model.
^{h} DRL - Design Representation Language.
^{i} Comp. Model - Compute Model; Conne. Graphs - connectivity graphs.
^{j} S: Security (① - confidentiality; ② - integrity.)
^{k} SDV - simulink design verifier
^{l} P - Power

3.3.2 Reliability Aspects

The key drivers for the reliability aspect in today's designs are the recent industrial standards in different application domains such as IEC61508, ISO26262, IEC61511, IEC62279, IEC62061, RTCA/DO-254, IEC60601, etc. Integrated circuits used in high reliability applications, e.g., complying with high (Automotive) Safety Integrity Level - (A)SIL, must demonstrate low failure rates (modelled by FIT –Failures in Time) and high fault coverage (e.g., Single-Point Failure Metric (SPFM) and Latent Fault Metric (LFM)). These requirements ultimately mandate extra-functional validation efforts for reliability analysis, such as Failure Mode and Effects (Criticality) Analysis - FME(C)A and imply generalized use of methods and features, such as safety mechanisms, for error management. Functional safety is a property of the complete system rather than just a component property because it depends on the integrated operation of all sensors, actuators, control devices, and other integrated units. The goal is to reduce the residual risk associated with a functional failure of the target system below a threshold given by the assessment of severity, exposure, and controllability.

The dominant threats for reliability are, first, random hardware faults such as transient faults by radiation-induced single event effects or soft errors [76], i.e., a subject for Soft-Error Reliability (SER). Second, these are extreme operating conditions, electronic interference and intermittence of permanent faults by process or time-dependent variations, such as aging induced by Bias Temperature Instability (BTI) [77], where the latter is a subject for Life-Time Reliability (LTR). Reliability verification challenge is emphasized by the adoption of advanced nanoscale implementation technology nodes and high complexity of systems, utilizing tens or hundreds of complex microelectronic components and embedding large quantities of standard logic and memory. Moreover, these designs integrate IP cores from multiple design teams making reliability evaluation task to be scattered and complex. Initiatives such as RIIF (Reliability Information Interchange Format) [78], allow the formalization, specification and modeling of reliability properties for technology, circuits and systems.

Similar to other aspects, reliability in large complex electronic systems, e.g., safety-critical CPSs, may be tackled starting at high level of abstraction. System's fault tolerance is formally checked by using UPPAAL and timed automata models generated from AADL specifications [47]. HW design models and tools at such a level also enable verification of interference of several extra-functional design aspects [45]. There are research works relying on design soft-error reliability verification by fault-injection campaigns, e.g., [55], or formal analysis, e.g., error-correction code (ECC) based mechanisms against single-bit errors in memory elements [54]. Burlyaev and Fradet [48] propose a general approach to verify gate-level design transformations for reliability against single-event transients by soft errors that combines formal reasoning on execution traces. Thompto and Hoppe [49] and Kan et al. [50] focus on the RAS (Reliability, Availability and Serviceability) group of extra-functional aspects outlined by IBM for complex processor designs where embedded error protection mechanisms and designs intrinsic immunity (due to various masking) to errors is evaluated by fault injection. Vinco et al. [51, 52] propose extensions to system descriptions in the IP-EXACT format to enable multi-layer representation and simulation of several mutually influencing extra-functional aspects of smart system designs such as lifetime reliability, power and temperature. A complex approach to verification of multiple reliability concerns (soft errors, BTI, etc.) across layers in industrial CPS designs is proposed in [53] as a collaborative research result in the IMMORTAL project. Last but not least, addressing the need for reliability verification automation tools, in [56], authors propose a fully automated tool INFORMER to estimate memory reliability metrics by circuit-level simulations of failure mechanisms such as soft-errors and parametric failures.

The survey clearly shows that currently there is a very small number of works considering verification of reliability together with other aspects.

3.3.3 Timing Aspects

Functional temporal properties are essential part of sequential designs' specification that are often modelled for functional verification by Computational Tree Logic (CTL), applied for formal approaches, and Linear Temporal Logic (LTL) temporal assertions expressed arbitrarily, e.g., in Property Specification Language (PSL), System Verilog Assertion (SVA) or systematically, e.g., in Universal Verification Methodology (UVM). In the extra-functional context, these can be extended to specific requirements and properties such as: real-time (RT), performance, throughput, latency, on-chip communication time constraint, worst-case execution time constraints, etc. Several works have been widely studying these timing properties. Some researchers are mainly focused on generating timing properties to reduce the verification efforts, for example, state space and cost [60, 62, 68]. Instead, other works use the timing properties to assess whether the system under verification is correctly functioning or not [61, 65, 67]. In the following, we discuss state of the art for each timing aspect.

- A real-time system describes hardware and software systems subject to a real-time constraint, that ensures response within a specified time. The correctness of the function depends both on the correctness of the result and also the timeliness of the periods. In [60], an approach to verify the timed Petri-Net model is proposed. A non-instantaneous model is abstracted from the timed Petri-Net model in a hierarchical structure. The non-instantaneous model which is verified with a model-checking tool is used to reduce the state space of the timed Petri-Net model for verification with a satisfiability modulo theories solver [79, 80]. The timed Petri-Net is used to model the interacting relations of the software components and the binding relations between software and hardware in a certain period of time. Görgen et al. [68] introduce a tool called CONTREX to complement current activities in the area of predictable computing platforms and segregation mechanisms with techniques to compute real-time properties. CONTREX enables energy-efficient and cost-aware design through analysis and optimization of real-time constraint. The authors in [65] propose a method to combine real-rime constraint aspect of a model with Energy-aware Real-Time (ERT) behaviors of the model into UPPAAL for formal verification.

- Throughput is a measure of how many units of information a system can process in a given amount of time. In [69], a verification environment has been proposed to estimate the throughput of a SoC. The intention of the paper is to judge whether the verification system can handle SOC verification and provide the necessary performance in terms of speed and throughput. Khamis et al. [70] introduce a Universal Verification Methodology (UVM) environment to measure throughput of a NoC. UVM is a SystemVerilog class library explicitly designed to help and build modular reusable verification components and test-benches. It is an industry standard, so it is possible to acquire UVM IP from other sources and reuse them.

- Performance refers to the amount of work which is done during a process, for instance, executing instructions per second. In [62], a framework has been developed to analyze performance of a system design. The framework is based on stochastic modeling and simulation and it is applied on a set of NoC topologies. The methodology uses a selective abstraction concept to reduce complexity.

- When referring to hardware, latency is the time required for a hardware component to respond to a request made by another component. However, in the case of hardware, latency is sometimes referred to as the access time . In [61], an analysis tool is developed to work with the AADL models [81] to assure the correctness of a scheduling model that binds the relation of different components in a model.

- On-chip communication time constraints refer to the requirements on the start and end times of each task in a system critical path, which is the sequence of tasks that cannot be delayed. For instance, in [57] and [58], a framework has been proposed, which is based on a set of quality of service aware NoC architectures along with the analysis methodology including selected relevant metrics that enable an efficient trade-off between guarantees and overheads in mixed-criticality application scenarios. These architectures overcome the notion of strictly divided regions by allowing non-critical communication pass through the critical region, providing they do not utilize common router resources. Such problem formulation is relevant to facilitate the usage of NoC technology by safety-critical industries such as avionics.

- The worst-case execution time of a computational task is the maximum length of time the task could take to execute on a specific hardware platform. The designer of a system can employ techniques such as schedulability analysis to verify that the system responds fast enough [82]. For instance, Zimmermann et al. [67] present an approach to generate a virtual execution platform in SystemC to advance the development real-time embedded systems including early validation and verification. These virtual execution platforms allow the execution of embedded software with strict consideration of the underlying hardware platform configuration in order to reduce subsequent development costs and to allow a short time-to-market by tailoring and exploring distributed embedded hardware and software architectures.

- Last but not least, a few works also take into account dependencies between several extra-functional aspects. For instance, the works in [65,68] and [62] present the effect of optimizing timing properties (performance and latency) on power consumption and the study in [67] performs the effect of decreasing execution time on power consumption. Such analysis is mostly limited to two extra-functional aspects or neglected at all [59–61,83], while design timing constraints can strongly influence not only power consumption but reliability, security, availability, etc. as well as functional properties.

3.3.4 Power Aspects

In commercial flows, verification of the power aspect can be addressed relatively independently from the functional verification dimension. The power intent and detailed power modelling can be done starting at TLM or RTL with minimal interference with the HDL functional description, e.g., using the Accellera introduced Unified Power Format (UPF) employed for power-aware design verification automation by commercial tools especially with the latest UPF3.0 [84] or Cadence/Si2 Common Power Format CPF [85]. For the advanced device implementation technologies, power specification implies multi-voltage design with up to tens of power domains and may consider dynamic and adaptive voltage scaling.

In the recent research works, design verification against the power aspect is performed at different abstraction levels with a trade-off between speed and accuracy. Some works such as [63, 64, 74, 75] perform power analysis at system level targeting high simulation speed and the similar accuracy achievable at lower levels. In [63], the authors apply their approach to SRAM and AES encryption IPs and obtained a significant simulation speed-up in comparison to gate-level simulation with a high fidelity of the system-level power simulation. A promising software tool for power simulation in SystemC designs is the Powersim framework [64, 75]. In [64], a methodology to estimate the dissipation of energy in hardware at any level of abstraction is proposed. In [75], the authors propose a SystemC class library aimed at calculation of energy consumption of hardware described at system level. The work in [73] introduces a series of tools (PowerBrick (construct power library for standard cell library), PowerMixer (for RTL/gate-level estimator), PoweMixer ip (IP- based model builder), PowerDepot (estimate system-level power consumption)) which can be tightly linked and enable the power analysis from layout, gate-, RT-, IP- to system level with a good simulation speed while retaining high accuracy. The power aspect verification could benefit from a holistic multi-level modelling, e.g., [86] available for functional verification. Rafiev et al. [62] , Vinco et al. [51, 52] , Kang et al. [65] , Zimmermann et al. [67] , Görgen et al. [68] , are aiming at methodologies suitable for specific applications (such as cyber-physical system [65]).

This extra-functional aspect has a tight relation to the implementation technology assumed for the synthesis of the design model under verification. With planar bulk MOSFET technology known for exponential growth of the static leakage power for smaller device geometries and employment of FinFET and Tri-Gate-Transistors in the advanced technology nodes, the CMOS device parameters are essential for this analysis [87] .

3.3.5 Machine Learning Based Techniques

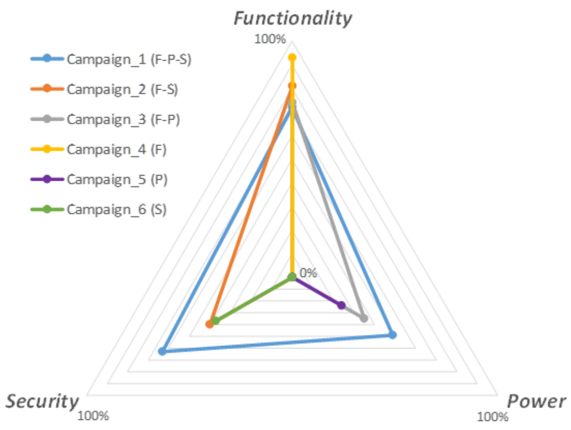
Machine learning has been efficient in solving classification, detection and design space exploration problems. For Electronic Design Automation (EDA), machine learning also shows potential to accelerate EDA procedures and problem-solving. As mentioned in [88], researchers have applied machine learning techniques to improve an EDA method, which can cover almost all abstraction levels in the integrate circuit design flow including design space reduction and exploration, logic synthesis, verification, etc. The machine learning discipline has shown the potential ability to solve the complex problem of multidimensional verification. This type of approach (along with e.g., evolutionary algorithms) is particularly suitable for multi-aspect optimization problems where formal deterministic approaches may lack scalability.

Machine Learning (ML) is the concept of a learning from examples and making predictions based on its experience, without being explicitly programmed [89]. Therefore, it can extract features or patterns from the target dataset. When applying ML for a design description, it can be used to extract circuit features at a high level and then use for other abstraction levels. Previous works have shown that ML can be used for verification purposes at different levels. In [90], machine learning is introduced in physical design analysis. The feasibility of ML in physical design verification (e.g., lithography hotspot detection) has been investigated, and a reference model for application has been presented. Based the work in [91], the applied machine learning method increases the speed of the performance evaluation (power and area) of a circuit design after physical design by a factor of 40. In [92], ML is used to predict the timing behavior of the final floor-plan of a circuit during the Place & Route routine and thus, shifting the analysis to an earlier design stage. In [93], the analysis is moved even to a higher design abstraction level. The High-Level Synthesis (HLS) resource usage and timing estimation are improved by training ML models with data from real implementations. Thus, the design flow can be assisted by machine learning and then predict accurate values even in very early design stages. Afterwards, Machine learning has been further applied for security verification in [94–96], where it is used to detect hardware Trojans based on features extracted from the gate level netlist. In Section 3.5, we propose an approach to assist the multidimensional verification flow by using machine learning techniques to estimate a reliability metric, as well as timing metric.

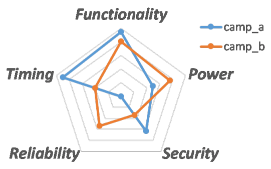
3.4 The Challenges of Multidimensional Verification

The performed analysis of the state of the art has outlined a gap in methodologies and tools for holistic multidimensional verification of hardware design models.

Different from functional verification, approaches for extra-functional hardware design aspects' verification remain underdeveloped even when tackled in isolation. Here, one of the key issues is a lack of established metrics for verification confidence. For a particular functional verification plan, the functional dimension usually includes conventional structural (code) coverage metrics, functional coverage [97] in form of asserted and assumed properties and design parameters along with stimuli quality assessment by model mutations [98]. The metrics for confidence in extra-functional dimension verification results may be challenging as in practice the requirements are subjective and can be specified as a mixture of quantitative and qualitative constraints. Accurate hardware verification in a particular dimension requires both sufficient design modeling and the targeted extra-functional aspects modeling [45]. There is a limited number of dedicated commercial tools and common standards for extra-functional verification flows. In particular, for the security dimension, the JasperGold SPV [46] is one of the few such commercial tools that stands out from the academic research frameworks. However, it only targets the specific security problems (sensitive information leakage and taints propagation through information flow in SoC design). Finally, the issue of eliciting the extra-functional requirements [99, 100] is a challenging task as ambiguity and (sometimes conflicting) inter-dependency of the extra-functional aspects in the specifications increase complexity of designing a general model to verify all the



(a)



(b)

Figure 7: Multidimensional Verification Campaigns (Radar-Chart n-Dimensional Visualization)

aspects and may leave gaps in the multidimensional verification plans.

Unfortunately, there is no established hardware design methodology supporting multidimensional verification plans for mutually influencing functional and extra-functional aspects. A very limited number of research works go beyond the analysis of one extra-functional verification aspect under constraints of another one, as the complexity of the problem grows extremely fast with the number of dimensions (interdependent constraints) and the size of the related electronic system. Research works in this direction are presented in Vinco et al. [52] and Vain et al. [45].

Ultimately, results of multidimensional verification campaigns proposed in this work are subject to be represented in a multidimensional space which needs to take into consideration different aspects both individually and interactively. The primitive concept of multidimensional verification is illustrated in Figure 7a. Here is shown an illustration of six hypothetical independent verification campaigns in a three-dimensional verification space. A verification campaign in this example shows the level of confidence in the different dimensions - (F)unctionality, (P)ower and (S)ecurity. In this illustrative example, only three aspects are taken into consideration. Obviously, on the demand of design requirements, the verification engineers can involve different dimensions. In the figure, the different colors of the lines represent different multi-dimensional spaces e.g., as Campaign_1 in blue lines stand for the verification result considering three extra functional aspects i.e., functional, power and security aspects at the same time. Subsequently, Compaign_2 represents the combination of functional and security aspects, Compaign_3 demonstrates the combination of functional and power aspect, etc. Thus the Radar-charts, as shown in Figure 7b, are an instrument for summarizing multidimensional verification results for a large number of dimensions, (where the dimensions can be ordered to emphasize correlation or inter-dependencies between adjacent dimensions).

3.4.1 Motivational Example

Single-dimensional verification campaigns, which ignore interdependencies among the dimensions, may lead to gaps in the overall electronic system quality. In order to shown the importance and necessity of the multi-dimensional verification, in this section we consider an actual verification campaign of an open-source NoC framework Bonfire [101, 102] as an example.

The design under verification is a 2 × 2 NoC infrastructure (processing elements excluded) implemented in RTL VHDL. It applies the open-source framework, which considers dependability constraints and is guided by a system health status monitor unit in the NoC in order to deal with task mixed-critical and non-critical application deployment. The verification plan considers 2-dimensional verification campaign targeting functionality and power consumption requirements. For the former, assertion-based functional verification by simulation is employed targeting statement, branch, condition and toggle coverage metrics and satisfaction of a set of temporal simple-subset Property Specification Language (PSL) assertions. For the latter, a set of power targets are extracted for the targeted silicon implementation assuming a particular switching activity (set to 12 mW in this example).

Among documented design errors in the Bonfire project, the bug f1, as shown in Listing. 1, is an example of a functional misbehavior due to improper usage of write and read pointers in the FIFO. The figure represents the code errors in the red line and the corrected versions of the code lines in blue. The bug f1 and the bug p1 demonstrate the error in Listings.1 and 2, respectively. The bug p1 causes violations of specified power consumption targets because of unnecessary excessive use of a fault-tolerance structure related counter.

The report of such a power consumption is described in Table 2. The power consumption is shown in the cell Total Power which is composed of the dynamic power, i.e., the Switching Power in the interconnects and the Internal Power in the logic cells, and the insignificant (for the target technology) static leakage power Leak Power. As summarized in the first row, for the bug f1 the Total Power is equal to 10.211 (consistence with the power consumption requirement). Similarly, in the third row, which represents the power consumption for the correct version of the code, the total power is equal to 10.184. It shows that even if there is a bug (bug f1) in the code but still the power consumption requirement is met. This fact indicates the design dependencies between functional and power aspects. However, when looking at bug p1 solo, the overall power consumption, which is 22.137, is about one time higher than the power requirement and thus it violates the power targets in the specification. But for functional requirement, there are no functional errors. In this sense, it is really difficult to locate the bug since it is really difficult to trace back. Thus, it is critical to know how and where the code should be modified to reduce the power consumption as well as maintain functional correctness. This fact shows the design inter-dependence between power and functional aspects. In general, the above simple motivation example demonstrates the challenge of inter-dependency of different aspects when requirements for more than one dimension are present.

Table 2: Power Consumption of the Bonfire System Implementation: corrected and with bugs f1 and p1

Bonfire system Implementation	Interconnects Power (mW)	Internal Power (mW)	Leak Power (pW)	Total Power (mW)
with f1 bug	0.783	9.427	7.50e+05	10.211
with p1 bug	0.757	21.379	6.93e+05	22.137
corrected	0.666	9.518	7.43e+05	10.184

```vhdl
process(write_en, write_pointer) begin --write pointer bug
  if write_en = '1' then
  write_pointer_in  <= write_pointer(0) & write_pointer(3 downto 1);-- Bug f1!
  else
    write_pointer_in <= write_pointer;
  end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer bug
  if (read_en = '1' and empty = '0') then
    read_pointer_in  <= read_pointer(0)&read_pointer(3 downto 1);-- Bug f1!
  else
    read_pointer_in <= read_pointer;
  end if;
end process;
```

```vhdl
process(write_en, write_pointer )begin --write pointer
  if write_en = '1' then
    write_pointer_in  <= write_pointer(2 downto 0)&write_pointer(3);
  else
    write_pointer_in <= write_pointer;
end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer
  if (read_en = '1' and empty = '0') then
    read_pointer_in  <= read_pointer(2 downto 0)&read_pointer(3);
  else
    read_pointer_in <= read_pointer;
  end if;
end process;
```

Listing 1: Bug f1 and Its Correction

```
process(Healthy_packet, reset_counters,
↪  healthy_counter_out,faulty_counter_out) begin
  if reset_counters  = '1' then
      healthy_counter_in <=  (others => '0');
  elsif Healthy_packet = '1' and  faulty_counter_out /= std_logic_vector(
  ↪  to_unsigned(0, faulty_counter_out'length)) then --Bug p1!
      healthy_counter_in <= healthy_counter_out + 1;
  else
      healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

```
process(Healthy_packet, reset_counters, healthy_counter_out) begin
  if reset_counters  = '1' then
      healthy_counter_in <=  (others => '0');
 elsif  Healthy_packet = '1'  then
      healthy_counter_in <= healthy_counter_out + 1;
  else
      healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

Listing 2: Bug p1 and Its Correction

3.5 Machine Learning to Tackle the Challenges of Multidimensional Verification

As proven in the previous sections, the verification problem is much more complex when the design under verification has more than one aspect requirements. In fact the complexity of verification, grows exponentially because of the interdependencies among the involved aspects.

Machine learning algorithms are known to be able to learn complex relationships and have been used for several optimization problems. Section 3.3.5 has shown that machine learning techniques has been already successfully used for estimating several different single verification metrics. This enables the assumption that machine learning can be also used for solving multidimensional verification problem. Therefore, in this section, we propose an initial machine learning based technique used for tackling the multidimensional verification challenge and show one possible solution toward multidimensional verification.

3.5.1 Proposed Methodology

The proposed approach targets at predicting two different verification metrics, timing and reliability aspects, based on the same feature set extracted from the gate level netlist of a given circuit.

These two different metrics are targeting at the design' de-rating and path delay prediction. The first metric to predict is the de-rating or vulnerability factor, which is related to the reliability verification flow and it is a major metric of the failure analysis. The second metric is the path delay and related to the timing analysis. This metric is usually obtained during the synthesis or place and route stage of the design development which can shift the timing aspect verification to the earlier design stages.

A possible application scenario consists of two steps. The first step is to extract a set of design features from the circuit. Then in the second step, it uses these limited set of reference input (the values of the selected circuit features) to train a ML model until the model can generate expected outputs (in this case, the output should be reliability and timing metrics). Depending on the exhaustiveness of the training campaign, the trained ML inference engine can provide actual reliability and time metrics from a limited list of circuit features while spending far less resources (CPU time, EDA tools licenses, man-power) than using classical methods.

3.5.2 Prediction Results

The proposed idea is implemented and evaluated on a practical example. The first step is to extract targeted design features. We select flip-flop instances in the circuit as the carriers of the targeted features. Then we focus on the behaviors of these flip-flop instances during simulation, and try to extract the feature-set based on the analysis of the behavior. The feature-set is composed of static elements (cell properties, circuit structure, synthesis attributes) and dynamic elements (signal activity). After extracting the features for the full list of circuit instances, reference data is obtained. For the second step, we use the reference data to train the machine learning models in order to get the correct prediction of the timing and reliability of the circuit. The reliability related functional de-rating for each flip-flop is determined through first-principles fault simulation approaches while the timing related path delay is extracted through a classical static timing analysis.

For the design under verification, the RTL design of the Ethernet 10GE MAC Core from OpenCores is used. The circuit consists of control logic, state machines, FIFO controllers, and memory interfaces. Then we synthesize the design with NanGate FreePDK45 Open Cell Library. In total, 1054 flip-flops are identified from the design. Afterwards, we have run the mentioned two steps to obtain the corresponding features.

For the first step, we simulate all the input references and collect the de-rating and timing information as input data sets for machine learning models. Among all the data sets, we have used part of the data sets as training data to train the machine learning networks and the other part as evaluating data to evaluate the correctness of the model. For the training procedure, several machine learning models have been used, such as the Linear Least Squares, Ridge (with linear and non-kernels), k-Nearest Neighbors (k-NN), Classification and Regression Trees (CART) and Support Vector Regression (SVR, with linear and non-linear kernels). The final result proves that the linear model is not suitable to predict the reliability metrics while the non-linear models perform much better. But the Support Vector Regression with Radial Basis Function (RBF) as kernel functions is among the best.

Therefore, the SVR model with RBF kernel function is used for the following presentation of the prediction results. The Figures 8 and 9 show the prediction of the two metrics. 50% (527 flip-flops) of the data are used to train the model and the remaining 50% are used to evaluate the model.

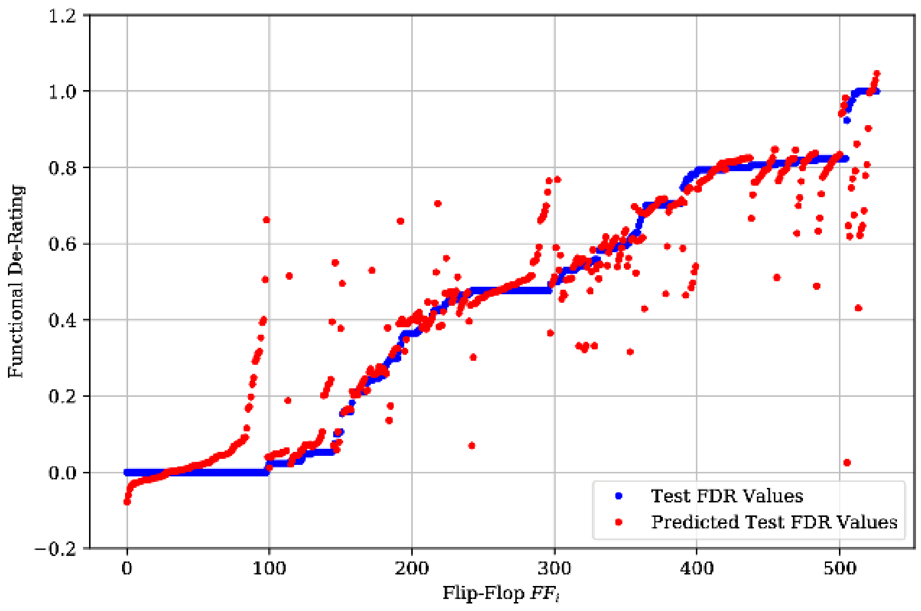


Figure 8: Prediction of Functional De-Rating Factors of the Test Data Set by Using A Support Vector Machine Regression Model (Training Size = 50%, Coefficient of Determination R2 = 0.844).

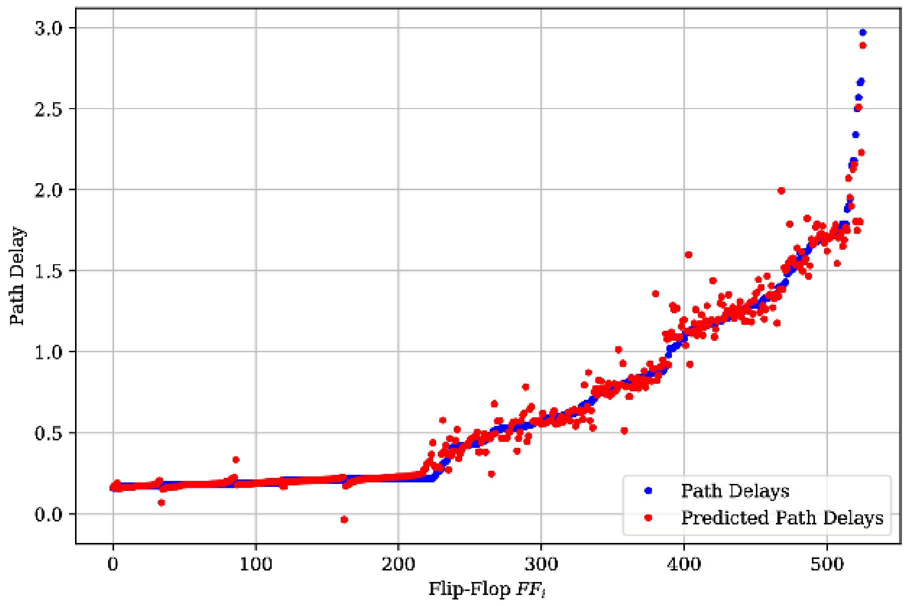


Figure 9: Prediction of Path Delays of the Test Data Set by Using A Support Vector Machine Regression Model (Training Size = 50%, Coefficient of Determination R2 = 0.975).

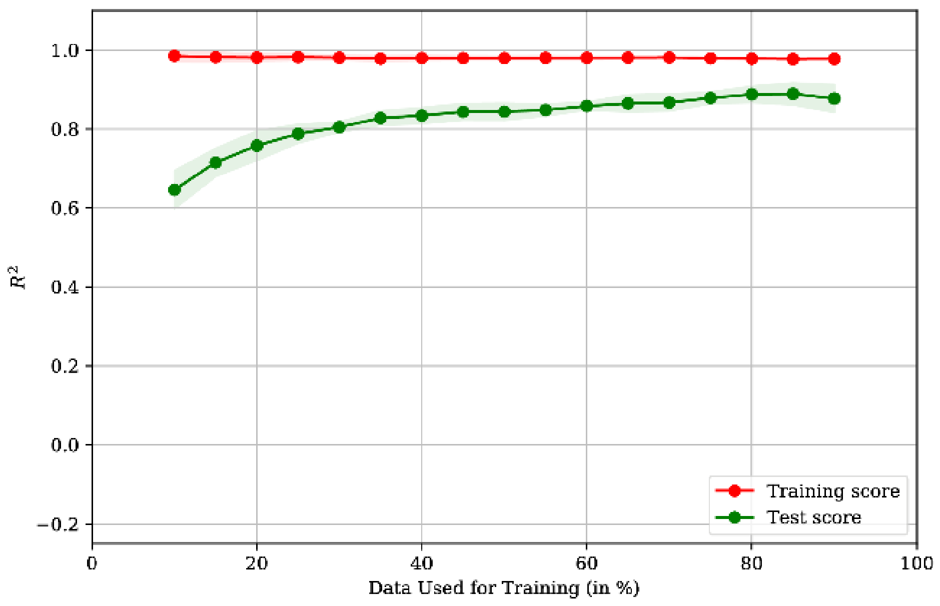


Figure 10: Learning Curve for the Functional De-Rating Prediction by Using A Support Vector Machine Regression Model with Different Training Sizes.

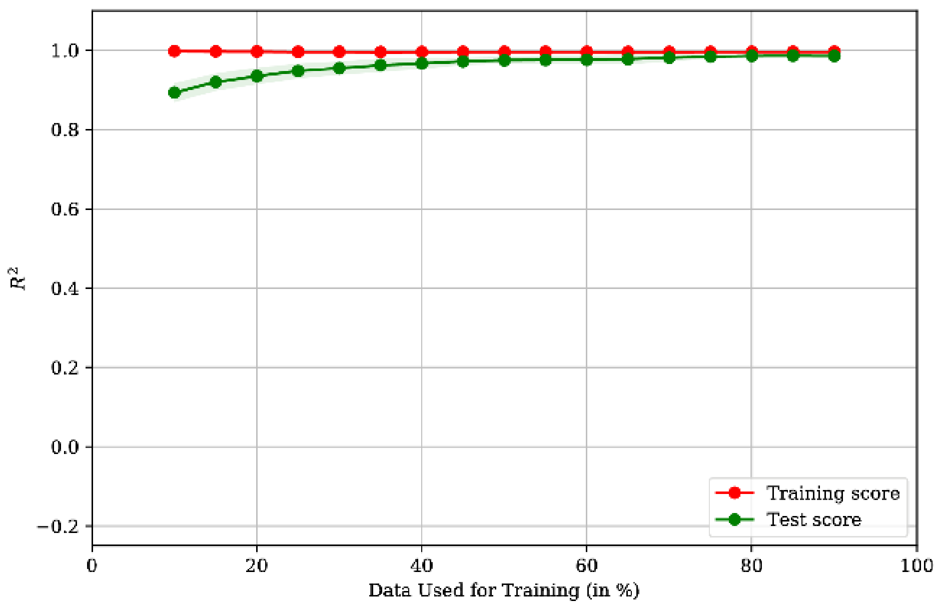


Figure 11: Learning Curve for the Path Delay Prediction by Using A Support Vector Machine Regression Model with Different Training Sizes.

The performance of regression models is evaluated by the Coefficient of Determination (R2) score. For the reliability related functional de-rating, the model reaches a score of

R2 = 0.844 while for timing related path delay, it reaches a score of R2 = 0.975.

For Figure 10 and 11, it shows the learning curve of the model. This curve describes the performance of the model for different sizes of training and evaluation data set. The learning curves suggest that it doesn't significantly improve the prediction performance by using more than 50% of the available data for training . However, it can also be seen that by using less than 50% of the data set, a valuable prediction still can be performed. Thus, by allowing a slight reduction of accuracy, the cost of an exhaustive analysis can still be reduced.

The practical example has shown that machine learning can be successfully applied for different verification purposes. In order use ML to support the multidimensional verification problem, features from different design stages need to be extracted and used to train a unified model or several separated models. These can be used to predict the required verification metrics.

3.6 Chapter Conclusions

In recent years, numerous extra-functional aspects of electronic systems are brought to the front and imply verification of hardware design models in multidimensional space along with the functional concerns. In this chapter, targeting at understanding of this new verification paradigm, we have performed a comprehensive analysis of the state of the art and presented a taxonomy of extra-functional aspects for hardware design verification, an up-to-date survey of related research works and trends towards enabling the multidimensional verification concept and investigated the potential approach of machine learning based techniques to support the concept. As the result of the performed analysis of the state of the art, we have outlined gaps in methodologies and tools for holistic multidimensional verification for hardware design models and the key challenges.

4 Security Aspect Verification

As presented in Chapter 3, the security aspect plays an essential role in ensuring the trust, integrity, and authenticity of modern electronic systems. For hardware security, the secure cryptographic primitives which have brought a new paradigm in secure communication in the electronic system, and PUF which uses the device's inherent variability during the manufacturing process as electronic fingerprints to protect IP or as key generation, are widely used security mechanisms. The existing literature on hardware security is considering ad hoc threat models, defenses and metrics for evaluation [10]. As proved by Table 1 in Chapter 3, methods used for attacking and protecting the systems are diverse when targeting different electronic systems and security scenarios.

This chapter focuses on the timing side-channel vulnerability analysis, as one of the most popular security topics, for both the cryptographic IP core and PUF involved electronic systems. Specifically, in Section 4.1, it provides a brief introduction of side-channel related hardware security. Then Section 4.2 and Section 4.3 address the study of timing side-channel vulnerability analysis of hardware designs of RSA crypto core and fuzzy extractors used in memory-based PUF enabled devices respectively.

This chapter is based on the following publications:

- **III** X.Lai, M.Jenihhin, J.Raik, and K.Paul, "PASCAL: Timing SCA resistant design and verification flow," in 2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 239–242, IEEE, 2019
- **IV** X.Lai, M.Jenihhin, G.Selimis, S.Goossens, R.Maes, and K.Paul, "Early RTL analysis for SCA vulnerability in fuzzy extractors of memory-based PUF enabled devices," in 2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC), pp.16–21, IEEE, 2020

4.1 Introduction

Security is not a first-class citizen in (hardware) design and is rarely considered during design space exploration. Bugs or vulnerabilities can originate from design flaws, some of which can be fully eliminated after a complete verification. The goal of the adversary in a security-critical application, is to learn or corrupt the secret information that one has no legitimate access to, e.g., the classified data or secret keys. Novel attack vectors like side-channel analysis rely on design features, to build efficient exploits that undermine assumptions regarding the accessibility of internal secret or exploitable information in a computing system.

The side-channel attacks can efficiently pose security threats to a system with integrated cryptography. Most of the side-channel vulnerability analyses for hardware designs presented in literature tend to break robust cryptography operations algorithmically through physical parameters like power, electromagnetic and execution time. For example, timing driven side-channel attacks exploit timing differences in execution traces.

As opposed to physical Side-Channel Attacks (SCAs) like differential power attacks, etc., that require physical access to the computer system, timing SCA can be launched (relatively) easily on general purpose compute environments that contain a memory hierarchy or performance-enhanced microarchitecture features like speculative execution. One exploits the difference in timing that arises because of a micro architectural artifact (e.g., the difference in access times between a cache hit and a cache miss) to create a timing side channel. The key invariant in these attacks is that there are different timing paths that provide out of band information.

4.2 PASCAL: Timing SCA Resistant Design and Verification Flow

Encryption algorithms and techniques for securing data continue to be an active domain of research in academia and on industrial fronts. They are fundamental in ensuring secure communication among electronic systems. Hardware implementations of encryption algorithms are being increasingly applied as the security mechanism in electronic system since they are regarded as a more effective root of trust.

Rivest-Shamir-Adleman (RSA) is a public-key cryptography algorithm and widely used in practice for secure communication, both as software libraries and the hardware security mechanism [103]. The RSA algorithm is asymmetric and is typically used for specific security cases such as sending sensitive data over to an open network. One of the examples of RSA used for secure communication scenarios is shown in Figure 12. The two users presented in the figure have their own secrets keys and public keys. The public key is publicly accessible whereas the private key is kept only by the user. When the sender wants to send the private (secret) data to the receiver, the private data need to be encrypted with the receiver's public key. Then the encrypted text, ciphertext, can be transmitted to the open network where everyone especially the attacker has the access. Then as the receiver, when he receives the ciphertext, he can use his private key to decrypt it and then get the private data. However, other illegitimate users can not.

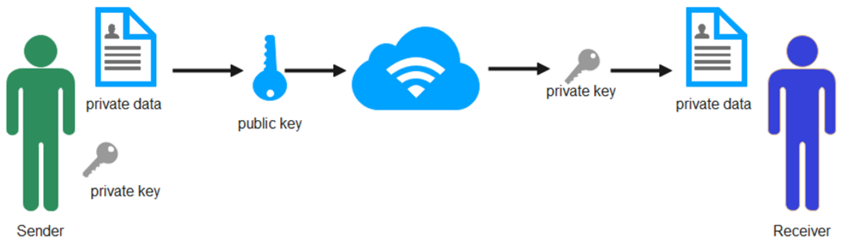


Figure 12: RSA for Secured Communication

As a public-key cryptographic algorithm, the RSA algorithm uses one-way function that is easy to compute in one direction but hard in the reverse direction. As mentioned before, the RSA algorithm involves two keys, the public key and the private key, which are mathematically linked. The generation of the RSA's public key includes a big number N generated by the multiplication of two large prime numbers. The security of RSA algorithm depends on the hackers' ability to factorize the big number N [104]. However, the reverse computation, factoring their products is difficult and time-consuming. Many researchers have studied the vulnerability of RSA system since its initial publication [105]. The best-known factoring attack is proven invalid for a typical key of more than 1024 bits in length for the conventional computer. However, recent research works present an efficient method, timing side-channel attack, to recover RSA private keys indirectly by observing its running time of the cryptographic algorithm and deducing the secret information.

4.2.1 Information Flow Tracking and Timing Side-Channels

Denning et. al. introduced the concept of secure information flow in a computer system whereby it can be shown that no unauthorized flow of information is possible due to control and data flow [106]. Consequently, Information Flow Tracking (IFT) has evolved and has been used as a formal methodology for modeling and reasoning about security properties related to integrity and confidentiality of side channels. The problem becomes more interesting and hard because high-level architecture abstractions are translated into transparent micro architecture implementations. While the hardware behavior in the micro-architecture can cause additional information flows which can be gainfully exploited to form these side channels.

One of the examples to merge IFT with timing side channels is the security path verification which addresses a specific, important aspect of overall security verification by checking access to the secure data in the hardware implementation to verify whether attackers can access the secure (secret) data through illegal logic paths. For example, in Figure 13, paths from A to B are controlled by the secret node S. Tools do taint propagation (i.e., taint analysis), which is a conservative approximation of secure information flow analysis, to find such paths [107]. A timing side channel exists if the contents of S can be derived/deduced by analyzing the arrival time of A at B. We call the two or more paths with unequal transit time as Timing Disparate Security Paths. Once we identify the set of Timing Disparate Security Paths, re-timing or balancing the paths for time can be undertaken to make the design robust with respect to timing SCA. Using Formal methods, it is possible to identify paths that allow taint propagation leading to information leakage. However, all such paths may not create a timing channel. We apply a commercial tool Cadence JasperGold SPV [46], to identify possible Timing Disparate Security Path candidates and then discard those paths that have equal execution traces. This helps us to enumerate all paths that could form a timing side channel and potentially be a vulnerability for a timing side-channel exploit.

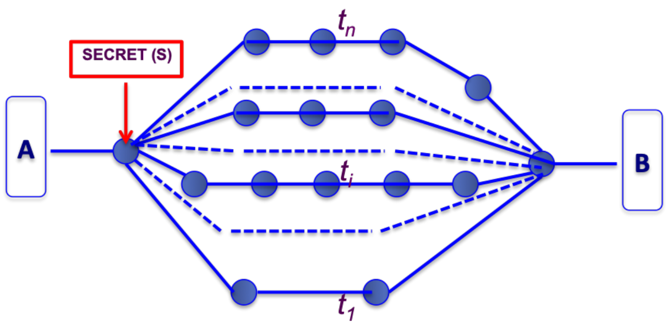


Figure 13: Timing Disparate Paths

The primary contribution of this research work is a secure automated digital design flow – PAth based Side Channel AnaLysis (PASCAL) – that can help to create a secure IP core or contribute to the system-on-chip security. The proposed flow starts from the RTL design and the threat model then uses a state of the art Security Path verification tool to identify potential timing side-channel vulnerabilities and last proposes a method to remove them by enforcing uniform timing to remove data-dependent instruction cycle count variations in the timing side channels.

The rest of this research work is organized as follows. Section 4.2.2 summarizes the state of the art in this area. Then Section 4.2.3 details the approach and presents the key algorithmic contribution in this work for identifying Timing Disparate Security Paths while Section 4.2.4 presents a lightweight method for Timing SCA Resistant Design using the results of the method proposed in the previous section. Section 4.2.5 describes the implementation results on a widely used crypto core and also demonstrates the efficacy of the proposed mitigation method. Section 4.2.6 summarises the contributions of the research work.

4.2.2 Related Works

Timing side channel attacks are known to be a hard and very important problem in modern systems. They have been used to extract cryptographic secrets from running systems [108]. Even differential privacy systems are not immune to these attacks [109]. And these are possible using both remote [110] and local adversaries [111]. Koeune et. al present an in-depth tutorial on side-channel attacks in [112].

A popular approach for defending against both local and remote timing attacks is to ensure that the low-level instruction sequence does not contain instructions whose performance depends on secret information. This can be enforced by manually re-writing the code, as was done in OpenSSL or by changing the compiler to ensure that the generated code has this property [113].

While methods for high-performance design or low power are available, design for security is still ad hoc. Only recently, systematic methods support design for trust and security have been described in literature [114, 115]. Menichelli et. al present an exploration approach centered on high level simulation based on SystemC to suggest improvements in the knowledge and identification of the weaknesses in cryptographic algorithm implementations [116]. Ardeshiricham et. al. have proposed an information flow based method for secure hardware design [117] by analyzing all logical code flows of the RTL code. In contrast, VeriCoq-IFT converts designs from HDL to Coq to analyze formal security properties [118]. SecVerilog requires explicit annotating each variable in the design with a security label — this is similar to using a type system to track information flow in the code [119]. Deng et. al have proposed a Computation Tree Logic to model execution paths of the processor cache logic and derive formulas for paths that can lead to timing side-channel vulnerabilities [120].

Most of the mitigation techniques that have been proposed try to remove data-dependent instruction cycle count variations by balancing timing or do a power flattening to remove power peaks/anomalies [121]. In some cases, Pipeline randomization for power and timing is also attempted. Alternatively, packet route randomization as a mechanism to increase NoC resilience has also been proposed [122]. Recently, Jiang et. al. have proposed a High-Level Synthesis (HLS) infrastructure that incorporates static information flow analysis to remove timing channels in a verifiable manner on HLS-generated hardware accelerators [123].

The methodology proposed in this work is based on a formal method that can identify all Timing Disparate Security Paths at RT level and improve the state of the art with a simple mitigation scheme for potential SCA vulnerable timing channels.

4.2.3 Methodology

The RSA algorithm has been shown to be vulnerable to timing SCAs and related mitigation techniques have also been proposed. However, the major focus continues to rely on verifying the correctness of encryption algorithms and their implementation in software and hardware. We present an approach based on RT level analysis that allows a precise understanding of possible flows for side channels based on timing. The methodology relies on a formal analysis tool Cadence JasperGold Security Path Verification App (JG SPV) [46]. The original objective of the tool is for security verification by checking access or leak of the secure data on the hardware to make certain that the attacker cannot breach the authentication logic and seek the secure data through illegal paths.

Based on a formal method of path sensitization from the secret information to the output observable points, we propose a method that detects possible Timing Disparate Paths in RTL designs that could be exploited as Timing Side Channel(s). As a result of this analysis, a simple and effective re-timing of timing SCA sensitive paths is proposed to make the design immune to the threats under the chosen threat model. We illustrate this on standard RSA RTL Verilog code.

Algorithm 1: Example: RSA Modulus Code

```
  Input: $C_m$,$K_n$; // C is the m bits cipher text, K is the n bits private key
  Output: $O_m$; // O is the m bits output plain text
1 $R_0 = Montgomery(C_m)$ and $R_1 = Montgomery(1)$ ;
2 $j = 0$ ;
3 while $j \leq n-1$ do
4     $R_0 = Montgomery\_Reduction(R_1 * R_1)$ ;
5     if $K_m == 1[j]$ then
6         $R_0 = Montgomery\_Reduction(R_0 * R_1)$ ;
7     end
8 end
9 $O_m = Montgomery^{-1}(R_0)$ ;
```

The decryption of the RSA modulus in Algorithm 1 uses Montgomery modular multiplication with the square-and-multiply algorithm. Here, we only focus on explaining the unintended timing channels in RSA, which attackers can use to reverse the private key. The details about how to choose the key or how the Montgomery algorithm works are out of this research's scope. In Algorithm 1, n, the bit number of K_n, decides total loop times while the value of single bit of K_n: $K_n[j]$ determines the operations for each single loop – only when $K_n[j]$ equals 1, statements at line 5 ,6, 7 will be executed while $K_n[j]$ equals to 0 will not. For the decryption of RSA, the total operations that need to be executed might be different with different private keys due to the above reasons. Assuming the time for single bit $K_n[j]$ is $t_{K_n[j]}$, the final execution time will be $t_{total} = \sum_{j=0}^{n} t_{K_n[j]}$. Thus, keys with a different number of '1's will cause a different execution time. This will open a timing side-channel for the attackers. Therefore, by doing a timing analysis, recovery of the value of K can be facilitated. The security of the RSA encryption scheme depends on the secrecy of the RSA modulus factors and, hence, this attack is potentially dangerous. It can be easily seen that there are potential n different timing classes that the observer will see at the output.

In order to verify this timing SCAs, we have proposed the verification flow: Timing Disparate Security Path Identification Flow (PASCAL) is shown in Figure 14 and the related algorithm is shown in Algorithm 2 separately. The tool used in this work to verify the secured IP is Cadence JasperGold Security Path Verification App (JG SPV) [46]. It is a commercial tool-set Cadence JasperGold Formal Verification applications and applied to RTL designs. It uses path sensitization technology and introduces a new type of property, path cover, which covers all the paths between the defined source and destination signals. This property addresses a specific, essential security scenario whether there is any unintentional path from/to the secured area, and this property can not be described by the conventional assertion-based formal method. It can be used to check the security information leakage and taint propagation in the system.

Algorithm 2: PASCAL Algorithm

```
   Input: Design Under Verification (DUV)
   Output: List of timing disparate Paths (P)
1  $cex_1$ = Execute JG SPV with DUV;
2  Append Path $P_1$ to P;
3  $i = 1$;
4  while $cex_i != 0$ do
5      Add $cex_i$ as new constraint ;
6      Initialize Counter;
7      $cex_{i+1}$ = Execute JG SPV with modified DUV;
8      Append Path $P_i$ to P;
9      $i = i + 1$;
10 end
```

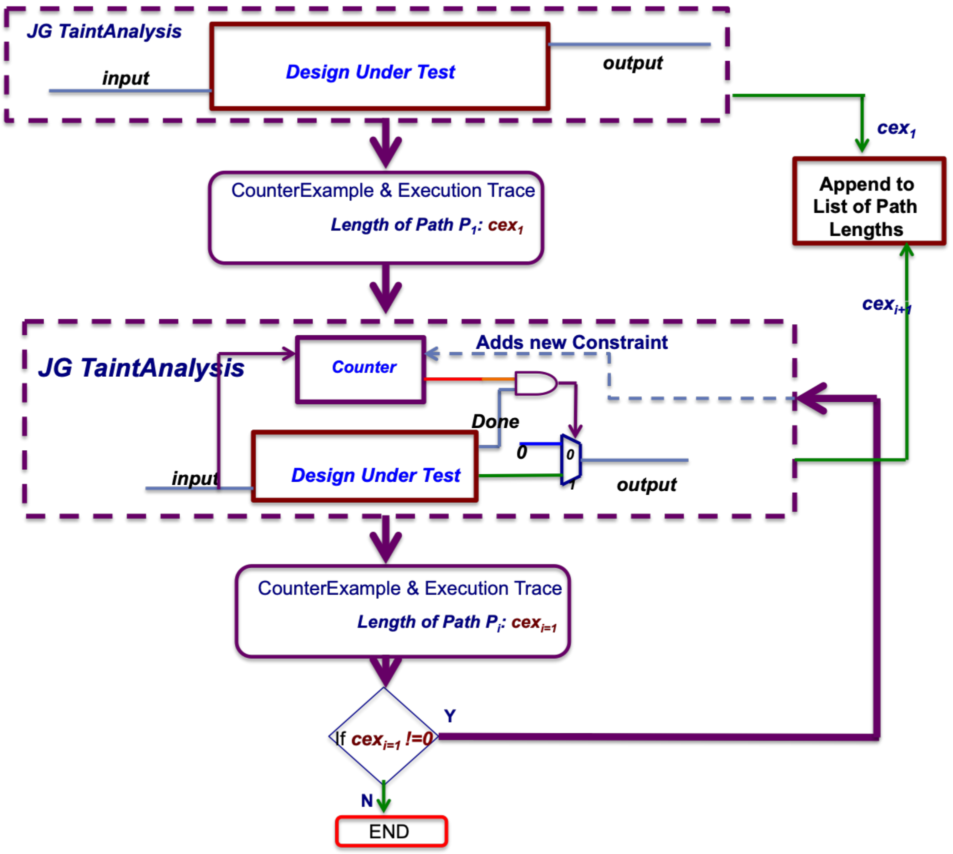


Figure 14: PASCAL Graphical Representation

In this work, we use JG SPV to analyze if there is one or several paths from a variable deemed to be secure and unobservable to the output. JG SPV uses a special path sensitization technology implying taint analysis to find if private key K can be propagated to the output O. As a first step, it generates a connected graph containing the states and variables and corresponding paths as shown in Figure 15. Next, the tool proves if there is a functional (i.e., sensitive) path among the ones in the initial set and if the path exists, JG SPV gives a counterexample as shown in Figure 16. The example shows the waveform of related signals along the path. We use the command "*[get_property_info -list{max_length} property_exponent_to_finish]*" to get the total execution time X (clock cycle based) of an existing path for this specified secure signal pair. Here it needs 44 clock cycles (additional 2 clock cycles are for set-up) to propagate.

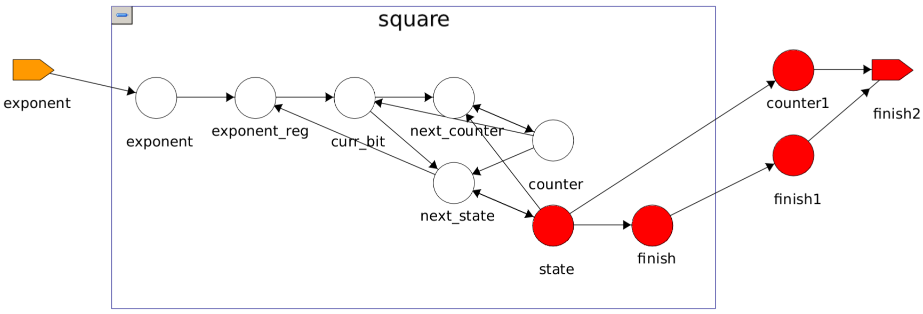


Figure 15: JG SPV Generated Connected Graph

After that, JG SPV is used to find another functional path (if it exists) from K to the output O with a time length different from X cycles. This is achieved by invoking JG SPV on a modified design, shown in Figure 17. A counter is added which drives the multiplexer to select the situation where the DUV both finishes the decryption *AND* also the length of the execution trace Y is not equal to X. If JG SPV finds another path with an execution trace length not equal to X and Y, it is added to the *Union Clause* P of the multiplexer select condition and the process is repeated until they find all the timing classes.

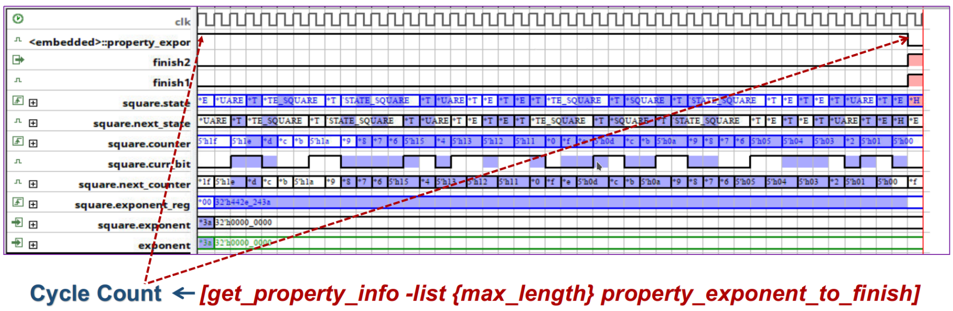


Figure 16: Counter Example and Execution Trace

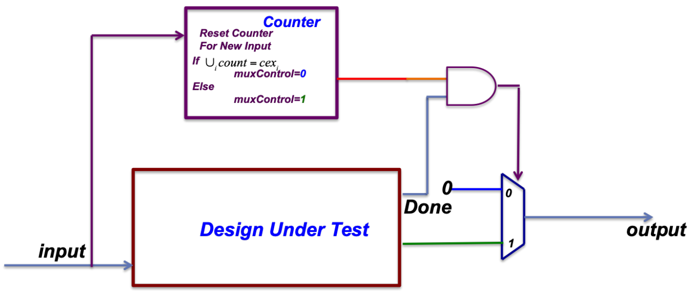


Figure 17: Modified DUV

4.2.4 Timing SCA Secure Design Flow

Considering the above timing side-channel vulnerability in RSA hardware implementation, this work also proposes a method that aims to achieve timing-sensitive noninterference for the synthesized design, via which it is ensured that confidential or secret values cannot be revealed by observing/measuring the timing of events at observable ports.

As explained earlier, the attacker can infer the value of the secret key by measuring the execution time of the RSA core. This is because of the existence of a strong correlation between the secret and the decryption time. An intuitive method to remove this Timing SCA vulnerability is to insert additional registers in the faster paths using path-balanced scheduling [124]. For example, if the two possible paths differ by 1 cycle, then a single register inserted in the slower path equalizes the timing and hence the timing sensitive vulnerability at the output port is removed.

However, as shown in Figure 18, there could be many paths $t_1, t_2, \cdots, t_n$ in the same basic block. Assuming there are $n/2$ paths each differing by one cycle, a path balanced scheduling synthesis procedure would insert $1+2+3+\cdots+n/2$ or $\mathcal{O}(n^2)$ registers.

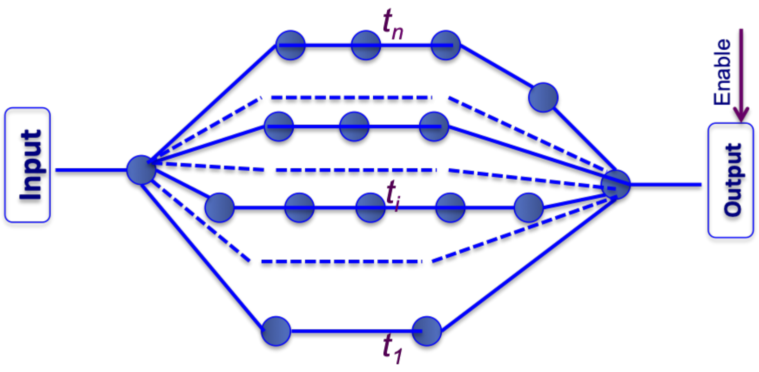


Figure 18: IP Core with Many Timing Disparate Security Paths

It may be noted that Timing Disparate Security Paths result in a Timing SCA vulnerability only if they are observable at user interfaces (output ports). This fact enables us to relax the constraints in the path balanced scheduling approach and enforce indistinguishable timing behaviors at the observable points in the design. Clearly, for the basic block or the core to be timing insensitive at the observable points, the output should be observable modulo t_{max} cycles where $t_{max} = Maximum(t_1, t_2, \cdots, t_n)$. As shown in Figure 19, we enable the output port/interface every t_{max} cycles using a counter and an AND gate. This small additional circuitry acts as the compensator

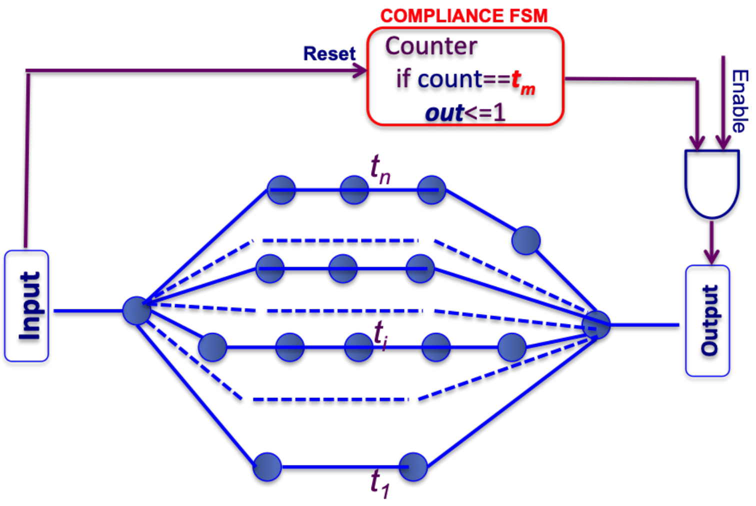


Figure 19: Proposed Mitigation Scheme

or balancing/compliance FSM and provides the (read) enable signal for observable interface. The contribution of the work is the automatic generation of the t_m parameter for the if statement in lightweight compensator from the Timing SCA Security Path Verification flow of the previous section.

This, therefore, leads to a very simple synthesis technique for ensuring a path balanced design with a single lightweight Compensator Block at the observable points of interest in the design. The additional circuit has a very small overhead counter which counts up to t_m to generate the control input for the AND gate which provides the enable signal to the observable register. The counter is reset every time a new input enters the basic block. This additional logic incurs no penalty in the critical path of the system — it has a performance impact because of path balancing as all paths now have the same latency. Algorithm 3 formally describes the method below. This method avoids resource duplication by having a uniform counter where the results from the different Timing Disparate Security Paths are delivered to the observable interface with the same latency.

Algorithm 3: Timing Channel Removal

```
  Input: Design Under Verification (DUV)
  Output: Secure Design Under Test (SDUV)
1 $P \leftarrow$ Output of PASCAL(DUV) ;
2 $t_m \leftarrow$ MaxExecutionLength(P);//Algorithm 2
3 *Compensator Logic Block* $\leftarrow$ Counter($t_m$) + ResetLogic;
4 SDUV $\leftarrow$ *Compensator Logic Block* + DUV
```

4.2.5 Experimental Results

The Montgomery modular multiplication with square-and-multiply algorithm based RSA cryptographic RTL implementation is vulnerable to timing SCA. This is because for different keys the time differences are dependent on the number of '1s' in the key as explained in Algorithm 1. In Figure 20, the time required to generate the timing disparate classes for 32-bits RSA, 64-bits RSA and 128-bits RSA are shown. For different RSA, the time needed to identify timing cases is varying: for the initial few time classes, they are obtained quickly while for the last few time classes they need a very long time.

Our method can correctly identify all timing classes using formal methods. i.e., for the 32-bits RSA verilog implementation, it identifies all the timing classes with cycle times from 33 to 64. As for the mitigation method mentioned in Figure 17. Since the counter needs to count to 64, we only need a 7 bits counter which incurs an approximate area penalty of 7 flops. In contrast, the path-balanced scheduling strategy would require about 512 flip flops. Clearly, with a 64-bit RSA, the savings are more significant. As mentioned earlier, the Compliance State Machine is not in the critical path and incurs no penalty in the operational speed of the circuit.

Since the counter has to count to 64, we need a 7 bit counter which incurs an approximate area penalty of 7 flops. In contrast, the path-balanced scheduling strategy would require about 512 flip flops. Clearly, with a 64-bit RSA, the savings are more significant. As mentioned earlier, the Compliance State Machine is not on the critical path and incurs no penalty in the operational speed of the circuit. The time required to generate the timing disparate classes for designs is shown in Table 3.

Table 3: Verification Time

Benchmark	No. of Classes	CPU Time (sec)
32 bit RSA	32	162
64 bit RSA	64	2565
128 bit RSA	128	38115

The initial timing classes are obtained quickly and as expected, it takes longer to find the last few classes as shown in Figure 20. The initial timing classes are obtained quickly and as expected, it takes longer to find the last few classes as shown in Figure 20.

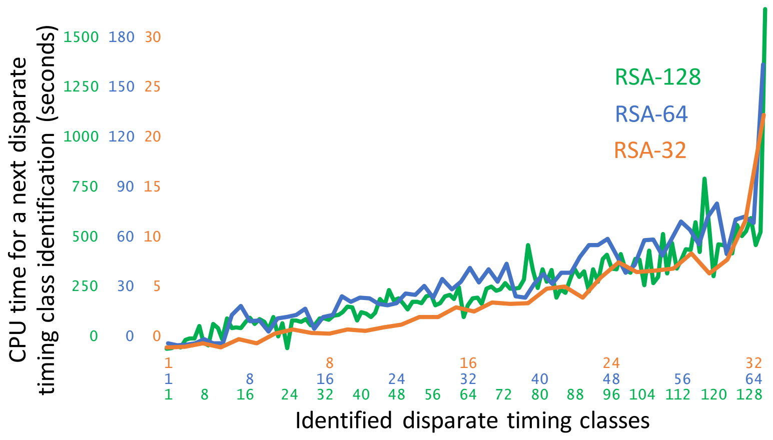


Figure 20: Normalized Execution Times

4.2.6 Conclusions

This section proposed a novel approach to discover timing SCA vulnerabilities in hardware designs and a lightweight mitigation technique to automatically eliminate the information leakage caused by the timing side channels. The proposed approach combines the information flow tracking with a formal method by using a commercial formal verification tool, Jasper Gold SPV. Compared with the state of the art in which the timing SCA is mostly addressed after manufacturing, this work is able to identify the timing SCA of the hardware design at the early design level and make it easier to fix, therefore reducing the design cost and the time-to-market. For the mitigation technique, the lightweight Compensator Block removes the timing side-channels with minimum modifications to the design and no impact on the combinational delay of the critical path in the circuit. The proposed approach and mitigation technique are demonstrated on a case study crypto core RSA hardware implementation.

4.3 Early RTL Analysis for SCA Vulnerability in Fuzzy Extractors of Memory-Based PUF Enable Devices

This section addresses the timing side-channel vulnerability analysis of the fuzzy extractor for memory-based PUF enabled integrated circuits.

Physical Unclonable Functions (PUFs) are hardware primitives which derive identifiers and cryptographic keys from the random variations of the silicon manufacturing process. PUFs provide a significantly higher security assurance as the generated keys are volatile and derived only when required. Thus, PUFs can be easily attached or embedded into the cryptographic implementation for authentication and identification [125]. PUF-enabled devices are also efficient alternatives to the expensive Non-Volatile Memory (NVM) to store the key because the keys generated by PUFs are derived by measurements in the field during the run time and don't need to be stored permanently.

However, PUFs are known to be sensitive to environmental factors such as the ambient temperature, the supply voltage noise, etc., which may affect the reliability of the response measurement, and ultimately, reduce the reproducibility of the cryptographic key. Along with the external factors, the internal factors of the PUF's manufacturing technology prevent it from guaranteeing a constant response all the time. This nondeterminism poses issues for applying a PUF as a key generator or identifier [126]. Therefore, for the post-processing, a Fuzzy Extractor (FE) is an essential component to help a PUF generate a reliable key by correcting the errors caused internally or by environmental variations.

Different types of PUF structure and the environmental conditions imply different requirements for the FE and the corresponding Error Correction Code (ECC). An example of a silicon PUF is the memory-based PUF, which is widely used in chip-level authentication. ECCs such as the Bose–Chaudhuri–Hocquenghem (BCH) [126] or Reed-Solomon [127] are usually used in memory-based PUF enabled devices. While FEs with ECCs significantly raise reliability, they can lead to new exploits such as allowing an attacker to extract sensitive information by studying the behavior of ECC. In recent research works, Side-Channel Attacks (SCAs) on ECC implementations have attracted particular attention of the research community. In [128], the authors extract the information about the key by non-invasive measurement of electromagnetic radiation together with a differential power analysis of the BCH decoder. In [129], the authors study the simple power analysis of both BCH and Reed-Solomon code and manage to recover the PUF response from the collected power traces.

However, there is no research work that refers to attacks that combine timing SCA and fault attacks for FEs, namely targeting to the execution time of the error-correcting code of FE in combination with the insertion of faults to PUF. So in this work, we address this gap by a study on BCH and Reed-Solomon RTL designs execution time differences as a reaction to intentionally triggered faults inserted into PUF. Specifically, the contributions of this study include:

- Definition of an attack model based on fault injection and timing analysis of ECC execution that may lead to the secret PUF values extraction.
- An early design stage RTL methodology for verification of an ECC design invulnerability against the proposed attack by employing both structural and simulation-based analysis steps.
- Case studies of Reed-Solomon and BCH based ECC with vulnerabilities identification and exploitation.

The rest of the section is organized as follows. Subsection 4.3.1 reviews the background and related works of the FE architecture and ECC decoders. The attack model is discussed in Subsection 4.3.2. Subsection 4.3.3 presents the proposed methodology for verifying invulnerability against the proposed attack. Subsection 4.3.4 presents a case study for ECC implementations and Subsection 4.3.5 concludes the section.

4.3.1 Background and Related Works

The Fuzzy Extractor [130] is a secure method to generate cryptographic keys from noisy sources. The FE serves as a post-processing unit in memory-based PUF-enabled cryptographic schemes. It is used both in the Generation and Reconstruction Procedures, as illustrated in Figure 21 and Figure 22 correspondingly.

In the Generation Procedure case, the fuzzy data from the PUF response W and a random secret S are used to generate the Helper Data by XOR operation on W and $E(S_0)$ which is encoded S_0. The generated helper data is stored in a non-volatile memory. In memory-based PUF-enabled devices, the Generation Procedure happens only once at the first-time power-on of the memory-based PUF.

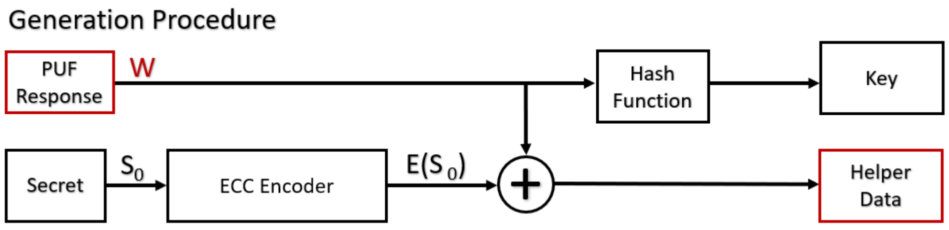


Figure 21: Generation Procedure in A PUF Fuzzy Extractor

On the contrary to this, the Reconstruction Procedure is executed many times during the product lifetime. Due to the noise and PUF manufacturing randomness, it is difficult to generate the same response consistently. To reproduce the correct cryptographic key, the Helper Data, stored in an NVM, is used in conjunction with the measured PUF response W'. Then with the help of the ECC decoder to detect and correct the divergent bits, the correct W is reproduced. After applying the Hash Function, the expected correct cryptographic key is reconstructed.

The FE guarantees that the resulting key is consistent while the publicly accessible Helper Data does not leak any information related to the secret of the key. To ensure

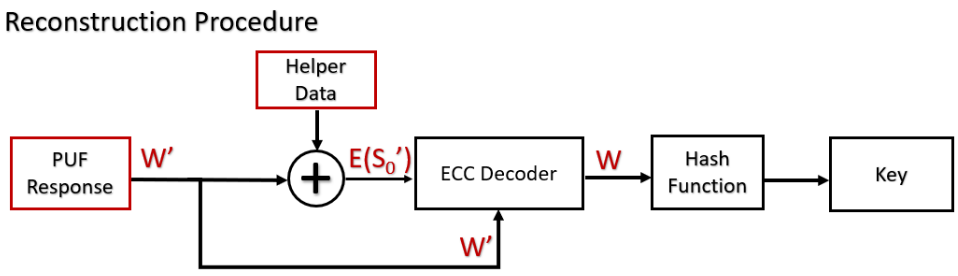


Figure 22: Reconstruction Procedure in A PUF Fuzzy Extractor

consistent generation of the correct key, the hamming distance between the measured PUF response W' with the originally measured W in the Generator Procedure should be smaller or equal to the correction capability of the ECC decoder, represented as a constant value $\boldsymbol{t}$. In this work, we assume that the measured responses of the memory-based PUF are within this hamming distance constraint.

Recent research works have identified potential attacks on FEs [131]. Most of them target the Reconstruction Procedure. In [132], the authors report on a method to extract the PUF secret by manipulating the Helper Data in the Reconstruction Procedure. In [133], Delvaux et al. provide an in-depth analysis of the Helper Data algorithms and identify new threats for leaking the Helper Data and the soft-decision coding.

The *ECC Decoder* is the main component in a FE. Binary BCH and Reed-Solomon are the two types of ECC that are widely used in PUF-enabled devices. Both codes are cyclic and capable of detecting up to $2t$ and correcting up to t errors by adding $2t$ check bits or non-binary values (symbols) to the data. Binary BCH is used for binary error correction, and Reed-Solomon is used for symbol error correction. While both software and hardware implementations of these codes exist, the hardware ones are more adopted. First, this is because the complex algorithms of the decoders require significant computational power along with the real-time constraints. The second difficulty for software implementations is the limited support of the Galois Fields Arithmetic operation in the general-purpose processors [134]. The hardware implementations of binary BCH and Reed-Solomon decoders are discussed in more detail in Section 4.3.4.

4.3.2 Attack Model

In this work, we assume an attack combining 1) fault injection to the memory-based PUF with 2) a timing SCA for observing and comparing the different decoding execution times of the ECC unit that is aimed at revealing the correct memory-based PUF data. In case of success, the attack explicitly compromises the core function of the PUF-enabled cryptographic devices, because the attacker can clone the PUF and can steal the secret.

Fault Injection Parameters For the physical fault injection to the memory-based PUF the following fault parameters are assumed.

(a) Granularity: each fault injection results in exactly one fault in one-bit data.

(b) Modification (fault type): after the fault injection, the manipulated data is set to a specified logic value, i.e., either '1' or '0'.

(c) Control: the attacker has a bit-wise precise control of fault injection to the memory-based PUF bits.

(d) Effect of the fault: the injected faults have a transient nature, i.e., the injected values are overwritten by the normal functionality of the device (e.g., the next measurement of the PUF on power-on).

Several studies on laser fault injection [135] have demonstrated similar attack parameters and, therefore, the feasibility of the above assumptions. Technical details of the fault injection attack implementation are out of the scope of this study.

Attack Assumptions The following set of assumptions must be satisfied for the success of the attack. The feasibility of the assumptions (iii)-(vi) is supported by several research works in state of the art.

(i) The output of a memory-based PUF measurement in the cryptographic device is processed by a FE with a binary BCH or Reed-Solomon based ECC.

(ii) The ECC implementation leaks exploitable information through the timing-side channel. *Comment:* The methodology for identifying the vulnerability enabling this assumption is the core contribution of this study and is presented in Section 4.3.3.

(iii) The memory-based PUF is noise-free under stable environmental conditions. The errors in the memory-based PUF are caused by the environment. *Comment:* While an ideal noise-free memory-based PUF would not require the FE at all, we assume that the noise is caused by the variations in the external environment while the internal noise is negligible. [136] demonstrates that the external environmental conditions like the ambient temperature, supply voltage, etc. have a significant impact on the error rate of the PUF.

(iv) The generated Helper Data is stored in NVM or the flash memory of the cryptographic devices and remains constant during the Reconstruction Procedure.

Comment: As an added value, this assumption creates an advantage for the proposed attack, compared to alternatives (e.g. [132, 133]), because it does not rely on the attacker being able to modify the Helper Data.

(v) The fault injection parameters (a) to (d) hold (see 4.3.2).

Comment: Several research works proposed bit-wise fault injection in SRAM and other on-chip memories. E.g., in [137], bit-wise faults were successfully injected into a PIC microcontroller through a semi-invasive method and without mechanical damage to the silicon.

(vi) The attacker has controlled access to measure the decoding execution time.

Comment: The physical measurement of the ECC decoding execution time can exploit the reflection of timing by the power traces. In [138], the authors analyze the use of the AES execution power traces for the SCA. The power traces are represented by changes of power over time, with the timing information embedded. A similar approach is used in [139] for RTL verification of RSA designs against vulnerability to timing SCAs.

Attack Procedure The proposed attack is a combination of fault injection with timing side channel analysis and represented by the following 4 steps. The procedure is illustrated in Figure 23.

1. Power on the device. Measure the initial PUF data. With the above assumptions, this memory value should be error-free, i.e., the same as W generated in the Generation Procedure. Measure and record the reference time T as the number of clock cycles for the execution of the ECC decoding.

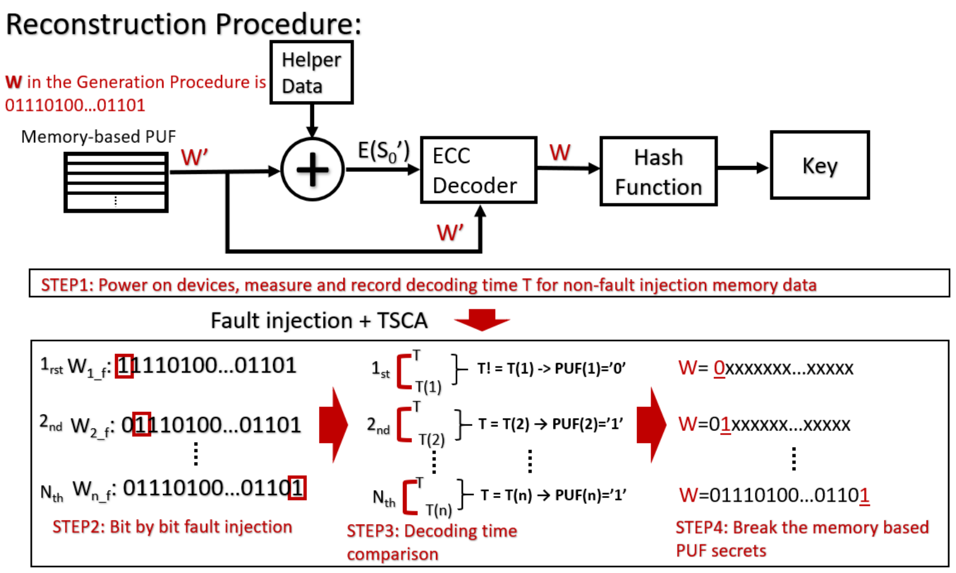

Reconstruction Procedure:
W in the Generation Procedure is 01110100...01101
Memory-based PUF
Helper Data
W'
E(S0')
ECC Decoder
W
Hash Function
Key
W'
STEP1: Power on devices, measure and record decoding time T for non-fault injection memory data
Fault injection + TSCA
1rst W1_f: 11110100...01101
2nd W2_f: 01110100...01101
Nth Wn_f: 01110100...01101
STEP2: Bit by bit fault injection
1st T, T(1) T! = T(1) -> PUF(1)='0'
2nd T, T(2) T = T(2) → PUF(2)='1'
Nth T, T(n) T = T(n) → PUF(n)='1'
STEP3: Decoding time comparison
W= 0xxxxxxx...xxxxx
W=01xxxxxx...xxxxx
W=01110100...01101
STEP4: Break the memory based PUF secrets

Figure 23: An Illustration of the Proposed Attack Procedure

2. Inject a fault f at the m_{th} bit of memory-based PUF following the (a) to (d) parameters and generate the new memory data W_{m_f}. W_{m_f} has a one-bit difference value compared to W. E.g, if the f is a set to logic "1" value and $m = 1$ then W and W_{1_f} can be either equal or can be different by exactly one bit at the first position. Then execute the Reconstruction Procedure, measure the decoding execution time $T(m)$.

3. The relation between these two decoding times T and $T(m)$ contains only two possible cases. The PUF's secret single bit m can be revealed by comparing the two decoding times as follows:

   - if $T! = T(m)$, then a different value at the m_{th} bit was injected. E.g., for $f = 1$, the original value of the m_{th} bit in memory is '0';
   - if $T = T(m)$ then the value at the m_{th} bit was equal to the injected one. E.g., for $f = 1$, the original value of the m_{th} bit in memory is '1';

4. Repeat the steps 1) to 3) of the procedure until the last m_{th} bit of memory-based PUF. The memory-based PUF's secret value is revealed.

4.3.3 Proposed Methodology

The precondition for the introduced attack is the non-constant decoding execution time in case of different input data for the ECC unit of the memory-based PUF Fuzzy

Extractor. In this work, we propose a methodology to identify this vulnerability in an ECC implementation already at the RTL design phase. The methodology employs both structural and simulation-based analysis for binary BCH and Reed-Solomon algorithms based hardware ECC implementations. In practice, these two algorithms are widely used by the industry in memory-based PUF-enabled devices.

Structural Analysis of ECC Decoder

1. Binary BCH Decoder: A general binary BCH decoder hardware implementation has three stages, as shown in Figure 24. The divergent (error) bits are identified by the Syndrome Calculator, Key Equation Solver and the Chien Search. Next, the decoder corrects the error bits by the XOR operation on the stored input with the identified error bits to recover the correct codeword. Let r(x), c(x) and e(x) be the received polynomial, codeword polynomial and error polynomial, i.e., r(x) = c(x) + e(x). Assume the binary BCH decoder can correct t errors. As the structural analysis of the binary BCH, we consider the following reasoning.

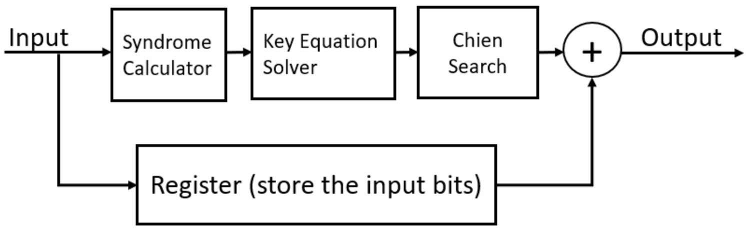


Figure 24: Binary BCH Decoder Structure

   (a) **Syndrome Calculator:** It is the first stage in the decoder generates 2t syndromes as defined in (1).

$$S_i = r(x^i) = r_0 + r_1 x^i + r_2 x^{2i} + ... + r_{n-1} x^{(n-1)i} \tag{1}$$

   where $1 \leq i \leq 2t-1$. An important feature of the syndromes is that they do not depend on transmitted information but only on error locations. If at position i there is an error then S_i has a non-zero value and it is equal to zero otherwise. For all possible inputs, the decoder always generates $2t$ syndromes. Therefore, the time for the syndrome calculation is constant for the BCH decoder with a fixed error correction capability.

   (b) **Key Equation Solver:** In the second stage, the error location polynomial $\sigma(x)$ is generated. Berlekamp Massey Algorithm (BMA) is one known iterative procedure that determines polynomial equation (2) out of a set of linear equations for the 2t syndromes calculated in the first stage.

$$\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + ...\sigma_t x^t \tag{2}$$

   BMA can be implemented in parallel or serially. In [140], it is demonstrated that a parallel implementation for errors correction BMA needs 2t iterations. A serial implementation implies a significant increase in the number of iterations. According to [141], it needs $2t^2$ iterations. However, for both cases, the total number of iterations is determined only by t, which is the maximum number of errors the decoder can correct.

(c) **Chien Search:** This stage searches for error locations by checking the roots of $\sigma(x)$. It is a simple trial-and-error procedure. All nonzero elements of the Galois Fields for a binary BCH decoder are generated in sequence and only capture the condition when $\sigma(x_i)$ is equal to zero which is the error position. Therefore, in this stage, the total number of nonzero elements depends only on the Galois Field GF(2^m) where $n = 2^m - 1$ and n is the size of codeword.

To conclude, for different binary BCH decoder implementations, the error correction bits and the size of the codeword are the factors which lead to the different decoding execution time. However, for a specific binary BCH decoder, these parameters are fixed at the design phase. Therefore, the structural analysis has not identified timing channels in binary BCH decoder structures.

2. Reed-Solomon Decoder: Reed-Solomon (RS) decoder aims at non-binary (symbol) error correction. Different from the binary BCH, which only needs to generate error locator polynomial $\sigma(x)$, RS also needs to generate an error value polynomial. Therefore, some RS implementations replace BMA with Euclidean Algorithm (EA) for the Key Equation Solver to calculate the error location polynomial and error value polynomial and add a new component Forney to calculate the error value. The Reed-Solomon decoder structure is illustrated in Figure 25. Here, the differences with the BCH decoder structure are highlighted in red. In the following structural analysis, we focus only on these two different components.

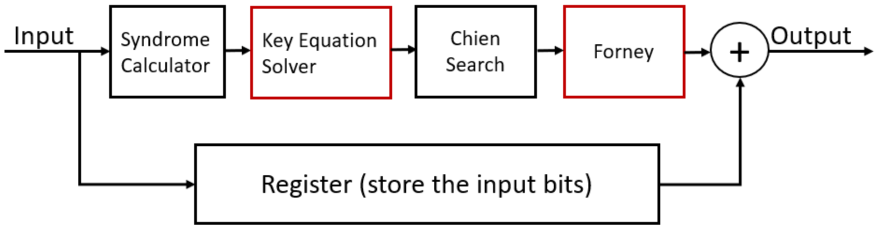


Figure 25: Reed-Solomon Decoder Structure

(a) **Euclidean Algorithm (EA):** It is an iterative procedure to generate the *error locator polynomial* and the *error value polynomial* with the 2t syndromes generated by the Syndrome Calculator stage. Particular implementations of EA may prefer a pipelined version with the objective of performance optimization [142]. In EA procedure [142], the error locator polynomial $\sigma(x)$ and the error value polynomial $\omega(x)$ are acquired by solving the equation (3). Equation (3) can be represented in the form of equation (4). The extend Euclidean Algorithm can find a series polynomial by (5). From (4) and (5), $A_i(x) = \sigma(x)$, $R_i(x) = \omega(x)$ and $B_i(x) = -Q(x)$. To solve the Key Equation the EA procedure starts with initiating the values $R_0(x) = x^{2t}$, $Q_0(x) = S(x)$, $L_0(x) = 0$, $U_0(x) = 1$ and then it is followed by interactions of four equations used to calculate $R_i(x)$, $Q_i(x)$, $L_i(x)$ and $U_i(x)$, based on the values from the previous stage, until the degree of $R_i(x)$ gets smaller than the degree of $L_i(x)$ or t. When the iteration is finished, the equation (3) is solved. Because the $R(x)$ starts at the degree 2t, and the iteration can finish at the degree of $R(x)$ equal to t or smaller. Therefore, the EA stage may require a different number of iterations for the different codewords which may introduce different execution times.

$$\omega(x) = S(x)\sigma(x) \mod x^{2t} \tag{3}$$

$$\sigma(x)S(x) = Q(x)x^{2t} + \omega(x) \tag{4}$$

$$A_i(x)S(x) + B_i(x)x^t = R_i(x) \tag{5}$$

(b) **Forney:** By using the Forney algorithm, the error value e(x) can be acquired by the equation (6).

$$e_j = -\frac{\omega(X_j)}{\sigma'(X_j)} \tag{6}$$

Normally, it is implemented in combinational logic because $\omega(X)$ and $\sigma(x)$ are available. The execution time of this stage is constant.

To conclude, the structural analysis has not identified the timing channel in the other stages of the Reed-Solomon structure but the second stage. Based on the implementation, the Key Equation Solver stage in the Reed-Solomon based ECC decoder can introduce the vulnerability.

Simulation-Based Analysis of ECC Decoder In an RTL simulation of an ECC decoder implementation, a number of stimuli data parameters may have an impact on the execution time of a decoding iteration. For the proposed simulation-based analysis step, the following parameters are identified:

- $codeword_{value}$: the encoded codeword value
- $error_{value}$: the error value is relevant only for a non-binary (symbol) ECC decoders
- $error_{position}$: the error bit position for a binary ECC decoder or the error symbol position for a non-binary ECC decoder
- $error_{number}$: the number of error bits or symbols for binary or non-binary ECC decoder correspondingly

The structural analysis of binary BCH and RS decoders and the defined attack model allows reducing the search space. Table 4 presents the relationship of the execution time variation introduced by manipulating a particular decoding parameter and the vulnerability to the proposed attack. The notations *C* and *NC* represent constant and non-constant decoding execution time, while *V* and *NV* represent vulnerability or invulnerability. In particular, manipulation of the $codeword_{value}$ parameter does not

Table 4: ECC Execution Time Variability and the SCA Vulnerability

ECC Decoding Execution Time/Vulnerability		
Parameters	RS decoder	Binary BCH decoder
$codeword_{value}$	C/NV	C/NV
$error_{value}$	C/V	
$error_{position}$	C/V	C/V
$error_{number}$	NC/V	C/V

identify the vulnerability of the target decoder. The attacker does not have access to manipulate the predefined correct codeword and can only manipulate the input codeword to cause an error. Based on the structural analysis, it is already known that different codewords do not introduce different decoding time neither in binary BCH nor in RS structures. The $error_{value}$ and $error_{position}$ parameters can be manipulated by the attacker by injecting faults to the input codeword. However, the constant decoding time will not leak information through the timing channel. From Table 4, we can conclude that the binary BCH decoder structures are secure with regards to the information leakage through the timing channel. An RS decoder implementation can be vulnerable if the attacker injects a different number of error symbols, i.e., the $error_{number}$. The table guides the designer which simulation campaigns are required to verify a particular implementation against vulnerability to the proposed SCA.

4.3.4 Case Study

The feasibility of the proposed methodology was validated by running an exhaustive simulation campaign on 3 case study ECC designs for memory-based PUF Fuzzy Extractors, i.e., 2 binary BCH and a Reed-Solomon ECC implementations.

Binary BCH Decoder The implementation of the binary BCH decoder is an open-source design in RTL Verilog accessible from Github [143]. Its general architecture is illustrated in Figure 24. The decoder was configured for a 12-bit codeword, 8-bit message and supports two types of BMA, i.e., serial *BMA_serial* and parallel *BMA_parallel* versions. The configuration was set to correct up to two errors, i.e., $t = 2$. Both versions were simulated with an exhaustive set of test vectors to identify the timing information leakage. Only valid values for the 12-bit binary codeword were extracted by running the encoder with all possible inputs. The input for the encoder is 4-bit message and 2-bit error correction capability. Since the number of errors correctable for a given polynomial is sparse, the encoder has the selection algorithm to select a suitable polynomial function to meet the provided requirements. Thus the actual message bit might be changed. In our case, the encoder pads 4-bit zeros and makes the input message bit 8-bit. We input all possible 4-bit value into the encoder. Then each encoded message value was merged with all possible error combinations considering the injection of 0, 1 or 2 errors at a time, i.e., all combinations of $error_{number}$ and $error_{position}$ were simulated. This means $T_{test\_vectors} = 2^4 * (\binom{12}{0} + \binom{12}{1} + \binom{12}{2})$=1,264 ECC decoding executions were analyzed for the each design, and the decoding time was measured.

Reed-Solomon Decoder The case-study Reed-Solomon decoder implementation is also an open-source design accessible from Github [144] and illustrated in Figure 25. The design was configured for 8-symbol codewords, 4-symbol messages and 8-bit symbols. The error correction capacity was also set to 2 errors, i.e., $t = 2$. By default, the design is pipelined by using registers to extend the execution time for each stage to the worst execution-time case. In practice, for memory-based PUF enabled devices where execution time is a critical factor, a configuration aimed at the decoder speed optimization is often used. This was also applied for the current case study. Different from the binary BCH, the Reed-Solomon decoder uses symbol-based error correction. While the parameter $error_{position}$ represents the position of the error symbol, the $error_{value}$ can take one of the $2^8 = 256$ possible values for an error in each symbol. The number of all combinations for the valid codewords merged with all possible errors for each symbol is $T_{test_vector} = \binom{8}{1} * (2^8 - 1) + \binom{8}{2} * (2^8 - 1) + \binom{8}{0}$= 1,822,741 that represents the number of executions to simulate and analyse per codeword. In the simulation campaign, we limited the analysis to one random valid codeword. Based on the architecture analysis, the other codewords provide the same results.

4.3.4.1 Experiment Results Analysis Experiment results are shown in Table 5. In the list of parameters identified for manipulation by the proposed methodology, the symbols "●" and "-" represent the varied and constant parameters correspondingly. T_d denotes the number of different decoding execution times identified and the corresponding values in clock cycles. For the Binary BCH, the experimental results confirm the conclusions of the structural analysis and do not identify any variations in the execution times. For the Reed-Solomon decoder, the red cells highlight the cases with the varying decoding time. In this experiment, *T_d:3 {38, 66, 72}* denotes different timing cases in case of the different number of errors to be corrected, i.e., 38, 66 or 72 clock cycles for 0, 1 or 2 errors correspondingly. As shown in the first three rows, different $error_{position}$ and $error_{value}$ can not affect the decoding time, and it remains constant (but can be equal to different values) $T_d : 1\ \{38\}\|\{66\}\|\{72\}$.

Table 5: ECC-based FE Decoding Timing Analysis

Varied Parameters				Decoding time by ECC Implementations (clock cycles)		
$codeword_{value}$	$error_{number}$	$error_{position}$	$error_{value}$	Binary BCH(BMA_serial)	Binary BCH(BMA_parallel)	Reed-Solomon-4-8-8
-	-	-	●			T_d:1 $\{38\}\|\{66\}\|\{72\}$
-	-	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	T_d:1 $\{38\}\|\{66\}\|\{72\}$
-	-	●	●			T_d:1 $\{38\}\|\{66\}\|\{72\}$
-	●	-	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	T_d:3 $\{38, 66, 72\}$
-	●	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	T_d:3 $\{38, 66, 72\}$
-	●	-	●			T_d:3 $\{38, 66, 72\}$
-	●	●	●			T_d:3 $\{38, 66, 72\}$
●	-	-	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	
●	●	-	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	
●	●	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	
●	-	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	

4.3.5 Conclusions

The application of Fuzzy Extractors for error correction may enable opportunities to break the secure PUFs if no countermeasures are taken. This work considers a combined attack model based on fault injection and timing analysis of ECC execution. In the worst case, such an attack may lead to the secret PUF value extraction. An early design stage RTL methodology was developed to verify the ECC design invulnerability against such or a similar SCA.

The methodology involves structural and simulation-based analysis parts. In our study, we targeted at two ECC architectures most widely used in FEs. The structural analysis has not identified vulnerabilities in the considered binary BCH architectures, while the architecture of Reed-Solomon based ECC may be vulnerable in particular implementations. A set of simulation-based experimental results have confirmed the findings and demonstrated the timing information leakage. Under the specified assumptions, the proposed attack procedure is able to exploit this vulnerability and reveal the secret.

The results of the early RTL analysis can guide in the selection of suitable ECC implementation or in the application of design-level countermeasures. To remove the leakage, e.g., a register can be added at the output of the Euclidean Algorithm stage to equalize the timing to the worst-case execution, or optimizations at the ECC algorithm may be applied. The efficiency of the mitigation solutions can be explored by the proposed methodology at a low cost.

4.4 Chapter Conclusions

In recent research works, the side-channel attacks are widely addressed and have been proven to be effective in posing security threats to electronic systems with integrated cryptography. Compared to conventional attack methods, side-channel attacks indirectly break robust cryptography operations through IC operation parameters like power, electromagnetic and execution time. Therefore, most of the research work has to carry out the side-channel attack' study after manufacturing which in return introduces additional design cost and increases the time-to-market.

This chapter discussed the timing side-channel attacks related security aspect verification of hardware designs with the focus on timing side-channel vulnerability analysis at the early design level. The work proposed both simulation-based and formal verification methods. The proposed hardware verification approaches are evaluated by case studies of timing side-channel vulnerability analysis for an RSA crypto core and a fuzzy extractor used in the PUF devices.

For the timing side-channel vulnerability analysis of RSA crypto core in 4.2, this work proposed a novel formal approach that combines information flow tracking with the formal verification method by using the formal verification tool JasperGold SPV. The proposed approach is able to automatically identify the exploitable timing side-channels for hardware designs and is evaluated by the case study of RSA crypto core. It successfully finds all the timing side-channels in the RSA hardware implementation. Then accordingly, a lightweight and efficient mitigation technique was proposed. The mentioned mitigation method introduces a small-overhead counter to the baseline design to equalize all the different timing paths at the observable output point and has no penalty in the critical path of the design.

For the analysis of fuzzy extractor used in the PUF devices in 4.3, this work firstly proposed a novel attack model which combines fault injection method and timing side-channel analysis of ECC used in fuzzy extractor. Then accordingly, it introduced a simulation based verification method to identify this vulnerability of the hardware design at RTL and the introduced method involves structural and simulation analyses. Validation of the proposed method has been carried out on three ECC decoders. The final results have demonstrated there is timing information leakages which can lead to the reveal of the secret PUF data in the hardware implementation of the Reed-Solomon decoder.

The above studies brought the timing side-channel vulnerability identification up to an early design level. The results of the early RTL analysis can reveal related design bugs in design phases which enable the design bugs to be fixed at the design level, therefore reducing the design cost and time-to-market.

5 Reliability Aspect Verification

In this chapter, the research work studies a Binarized Neural Network (BNN) hardware inference engine used in critical applications with certain levels of reliability and security requirements. The analysis focuses on the reliability study of the security enhanced BNN inference engine and explores the interdependent design space between security and reliability aspects.

This chapter is based on the following publication:

- **V** X.Lai, T.Lange, A.Balakrishnan, D.Alexandrescu, and M.Jenihhin, "On antagonism between side-channel security and soft-error reliability in BNN inference engines," in 2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1–6, 2021

5.1 On Antagonism Between Side-Channel Security and Soft-Error Reliability in BNN Inference Engine

Deep Neural Networks (DNNs) are designed to classify objects after training and have been proven effective in many Artificial Intelligence (AI) applications. To improve the energy efficiency and throughput, a trained DNN model can be mapped to a hardware inference engine [145]. As a rule, generating an industrial DNN model implies a significant amount of training data, computing resources as well as human efforts. Therefore, it is essential to protect a DNN model against data and functionality integrity violations, intellectual property rights and potential illegitimate reproduction [146]. On the other hand, the DNN inference engines are increasingly adopted for critical applications demanding high levels of functional safety and reliability to in-field faults.

A Binarized Neural Network (BNN) is a particular class of DNNs proposed in [147] and characterized by binary weights and activation functions that require less storage and computational resources compared to full-precision DNN models. Therefore, BNN hardware inference engines are efficient for applications in critical embedded systems [148], e.g., autonomous robotic vehicles and ML-powered edge devices. However, recent research works have discovered that the simpler binary operations in BNN have accidentally created power side-channels. In a recent research work [149], Anuj Dubey et al. show that the Differential Power Analysis (DPA) [150] on a running BNN hardware inference engine can extract the secret model parameters such as weights. Using the methodology described in [151], the authors repeatedly apply the DPA method on a BNN with 4-bit weight values. Based on 100k measurements, they compute the corresponding power consumption of the intermediate computation for all the 16 possible values and discover a significant correlation between the power measurements and the correct weight values, which can be exploited for information leakages. To address DPA in BNN hardware inference design, data masking techniques are proposed in [152]. In the paper, the authors propose a fully-masked BNN hardware inference engine design by masking all the linear and non-linear operations presented in the implementation and validate the effectiveness of the method on an FPGA with 2M power traces.

Recently, BNN hardware implementations are increasingly employed for critical applications with the requirement for in-field lifetime reliability even in the presence of hardware-level faults [153]. Unfortunately, the described above security enhancements modifying the logic structure of the design may negatively affect the soft-error reliability of the BNN. To the best of our knowledge, the effects of such side-channel masking techniques on soft-error reliability of the hardware design are not duly addressed in the literature.

In this work, we propose a soft-error reliability analysis of the security-enhanced BNN. We apply an existing masking technique to the BNN hardware implementation and prove that the reliability of the DPA-resistant HW BNN inference engine is compromised.

This research work makes the following main contributions:

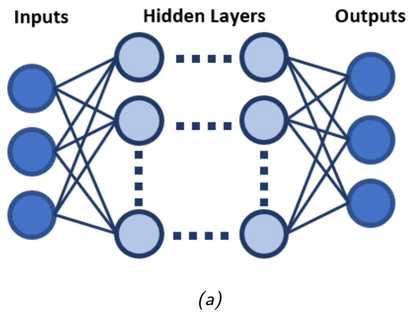
- Establishing a novel discussion for reliability jeopardy introduced by side-channel security countermeasures in the HW BNN inference engines.
- Proposing a soft-error reliability analysis flow for HW BNN inference engines based on logical and functional de-rating factors evaluation.
- Proving a significant increasing of the soft-error vulnerability in a representative case-study open-source BNN design, specifically, up to $1000\times$ output bitwise failure rate increase or up to $350\times$ neural network functional failure rate increase.

The rest of this chapter is organized as follows. Section 5.2 reviews the background of BNNs, the relevant power side-channel attacks and the corresponding mitigation techniques. In section 5.3, the soft-error reliability assessment is discussed. Section 5.4 explains the case-study BNN architecture and the performed reliability analysis, followed by experimental results in Section 5.5. Section 5.7 concludes this chapter.

5.2 Related Works

5.2.1 Binarized Neural Network

Generally, a BNN contains an input layer with input vectors, several hidden layers composed of internal operational neurons, and an output layer that computes the final result (see Figure 26a). In BNNs, the arithmetic computations use binarization, i.e., the most extreme form of network quantization [154]. Therefore, the weight multiplication can be replaced by a simple bitwise operation XNOR operation. The summation can be simplified by using a population count (or *popcount*) that calculates the number of '1' in a bit vector (see Figure 26b).



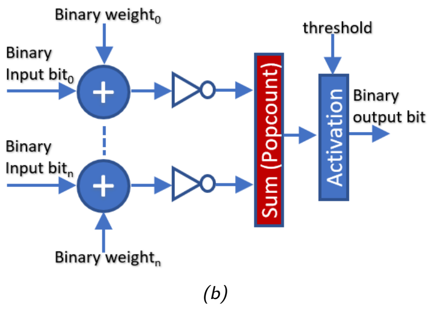


Figure 26: Binarized Neural Network

5.2.2 Power Side-Channel Attacks on BNNs

However, the simple structure of such a neural network has its drawbacks. Recent research works have demonstrated that BNN model can be exploited by physical side channels such as time, power and electromagnetic emanations [155–157]. Notably, the work in [149] presents an impressive Differential Power Analysis side-channel attack that can reveal the secret weights of a fully connected BNN. Considering that the weight value is binary, the summation is usually implemented by a pipelined adder tree composed of several stages that need additional registers in the middle to store intermediate values. The adversary can target any stage of the adder tree, primarily focusing on the switching activities of the registers in the adder tree to extract the power traces. With the power traces at hand, it is feasible to steal the weights and biases values of the trained model. In the mentioned example, the authors apply the DPA method [151] on the second stage registers of the adder tree and succeed in extracting the values of the weights and the biases for all nodes by analyzing the Pearson correlation coefficient of the power traces.

5.2.3 BNN Power Side-Channel Masking Techniques

To address the above vulnerability, [152] proposes techniques for the power side-channel masking aiming at enhancing the HW BNN inference engine's resistance to DPA. In the paper, the authors analyze the neural network specific computation and propose the following masking techniques:

- For the weighted summations, build a protected AND operation by adopting the glitch-resistant Trichina's AND gate for the Ripple-Carry Adders (RCA) to replace them with the security-masked N-bit RCA;
- For multiplexers, use Look-Up Table (LUT) as a replacement, e.g., an 1-bit multiplexer is replaced with a 4-input 2-output security-masked LUT.
- For the activation functions, perform the NOT operation by inverting one of the two Boolean shares of MSB received from the previous security-masked adder.
- For the output layer, transform the problem of the masked comparison to masked subtraction and reuse the masked adder as a subtractor.

Among the four mitigation techniques, only the first one, i.e., the RCA masking, introduces numerous extra registers, depending on the size of the adder. This particular technique creates an important case study for our analysis. Intuitively, the introduced massive sequences of flip flops behave as "magnets" to additional soft errors to be caught by the BNN in the field. According to our hypothesis, a BNN protected against security side-channels with this or a similar technique becomes significantly more vulnerable to the soft-error reliability issues. Interestingly, the adopted Trichina AND gates structure [158] is also known in the literature for its application to AES co-processors and similar cryptocores to protect them from power side-channel attacks. This fact potentially extends the impact of our analysis.

5.3 Soft-Error Reliability Assessment for HW BNN Implementations

5.3.1 De-Rating Based Soft-Error Reliability Assessment

With the technology feature size shrinking, the probability of electronic systems to experience Single Event Effects (SEEs) is increasing and the overall vulnerability to the radiation-induced transient faults, i.e., soft errors, is becoming more prominent [159]. To evaluate the reliability to soft errors for a circuit, the analysis of sensitivity of the underlying cell's technology is used. However, not all faults occurring in the cells lead to failures, i.e., observable effects at the application level, but can be masked on the way. De-rating factors are used to quantify the masking effects of soft-errors. At logic level, the de-rating analysis relies on logic-level models for SEE, i.e., a Single Event Upset (SEU) for sequential cells and a Single Event Transient (SET) for the gates. Moreover, in an RTL analysis, the SET faults, can be modeled with a reasonable accuracy by a subset of SEUs. The overall Soft-Error Reliability (SER) for SEUs in a system circuit can be expressed by Equation (8), where $\mathrm{SER}_{\mathrm{SEU},i}$ is the rate of SEUs in a sequential cell *i* (a flip-flop). $\mathrm{SER}_{\mathrm{SEU},i}$ is calculated by Equation (7) [160, 161], where the Failure In Time $\mathrm{FIT}_{\mathrm{SEU},i}$ denotes the rate of soft errors for the sequential element and depends on the underlying technology and the operating environment. The considered de-rating factors are Temporal De-Rating ($\mathrm{TDR_i}$), Logical De-Rating ($\mathrm{LDR_i}$), Functional De-Rating ($\mathrm{FDR_i}$), respectively.

$$\mathrm{SER}_{\mathrm{SEU},i} = \mathrm{FIT}_{\mathrm{SEU},i} \cdot \mathrm{TDR}_i \cdot \mathrm{LDR}_i \cdot \mathrm{FDR}_i \tag{7}$$

$$\mathrm{SER}_{\mathrm{SEU}} = \sum_{i \in \text{flip-flops}} \mathrm{SER}_{\mathrm{SEU},i} \tag{8}$$

5.3.2 Reliability Assessment for the HW BNN Implementations

Recent research works [153, 162–166] have demonstrated the significance of the soft-error reliability study for machine learning systems in safety-critical applications like autonomous unmanned robotic vehicles and AI-powered edge devices. One of the interesting questions in the DNN reliability research is related to understanding the resilience of particular layers to transient and permanent faults. The conclusion in [167] considering faults in registers of a RTL model is that permanent faults in inner layers cause less inference errors than the permanent faults in the first layers. This is in line with the studies of permanent faults in [159] and [168]. However, according to [167], the later layers are more vulnerable to transient faults compared to the first layers. Also the research in [169] and [170] demonstrates that the random bit flips in weights for the early layers has less impact on the inference accuracy compared to the faults injected to the last layers. In our current analysis, as discussed in next section, we follow the arguments that the *last layers* of a DNN are the most critical for the SER assessment.

The studies for DNN reliability assessment include frameworks, e.g., Fidelity [165], for abstracting the level of or optimizing the fault analysis for DNN hardware inference engines such as fixed-/floating point CNNs generated by NVDLA, Eyeriss, MAERI, FINN, CAFFEINE, ISAAC and similar generators. Frameworks for automated generation of binarized NN HW inference accelerators from established DNN evaluation and training platforms such as Tensorflow, PyTorch or Caffe are limited. Therefore, the SER analysis flow for the BNN designs (initial ones and the ones containing the security enhancements) employed in our work relies on a two-step approach. First, we perform de-rating evaluation for the critical parts of the case-study BNN, e.g., the security-masked ripple carry adder. Second, we execute a fault injection simulation campaign.

For the first step we rely on the SoCFIT tool by IROC. It is a reliability-focused design characterization tool that predicts various de-rating factors and calculates the failure rate of digital circuits. The tool computes the SEU logic de-rating by analyzing the structure of the RTL description of the circuit. Therefore, all paths from the sequential cell to any end point (primary output) are considered and the LDR for each found paths is evaluated, by multiplying the intrinsic LDR for each gate along the path. The overall LDR for the sequential cell can then be computed by either selecting the value of the path with the maximum path or averaging the results of the individual paths.

5.3.3 BNN-level Fault-Injection Simulation Campaign

In this work, the reliability analysis for HW BNN inference engine implementations is performed by statistical fault injection simulations on the RT level of the design. The BNN circuit is simulated with a logic-level simulator by running a corresponding inference testbench. With the testbench the correct behaviour of the BNN circuit can be verified. This can be achieved by monitoring and recording all outputs of the neural network or testing the functional behaviour of the inference engine. First, a simulation is run without any faults injected to create a reference golden run. Afterwards, a random target flip-flop is chosen where the fault is injected at a random time. The SEU in a flip-flop can be emulated in the simulator by inverting the value of the target flip-flop. This procedure is automated by using a standard state-of-the-art simulator and its routines and commands. In the presence of a fault, the simulation run is compared with the golden run and any differences at the output or any differences in the functional behaviour of the circuit can be observed and recorded.

5.4 Reliability Analysis of Secured BNN Hardware Inference Engine

5.4.1 Implementation of Power Side-Channel Masking in BNNs

As mentioned previously, several power side-channel masking techniques are proposed in the state of the art, e.g., [152]. In our analysis, we focus on the weighted summations for the soft-error reliability study according to our hypothesis.

Instead of an adder tree, which is used for the popcount implementation and is vulnerable to power side-channel attack [149], a ripple-carry style adder is recommended. For the normal 1-bit RCA, the logic functions for the summation and the carry-out bit are provided in (9) (10) respectively, where a, b, c, C and S are the two one-bit inputs, one-bit carry-in, carry-out and summation.

$$S = a \oplus b \oplus c; \tag{9}$$

$$C = a \cdot b \oplus b \cdot c \oplus a \cdot c; \tag{10}$$

Based on these equations, the complete addition computations are expressed as a sequence of XOR and AND operations. As indicated in [171], only the AND operation in the summation needs to be masked since it is a non-linear operation, while the XOR operation is a linear operation and, therefore, can be left unmasked. Among the recent related masking styles [172], the Trichina method is chosen for AND gate masking because of its simplicity and efficiency [152]. However, the straightforward adoption of Trichina's AND gate causes glitches in the circuit [173], which can lead to information leakage. Thus additional registers are introduced to stabilize the signals in the design. The glitch-resistant Trichina's AND gate is shown in Figure 27, where the secret variables (a and b) are split into different shares (a_0, a_1, b_0, b_1), r is a fresh random bit, and the additional registers are marked in red. The equations used for sum bits and carry bits are shown respectively in (11), (12), (13) and (14)).

$$S_0 = a_0 \oplus b_0 \oplus c_0; \tag{11}$$

$$S_1 = a_1 \oplus b_1 \oplus c_1; \tag{12}$$

$$C_0 = Tri_0(a_0, b_0, r_0) \oplus Tri_1(b_0, c_0, r_1) \oplus Tri_2(c_0, a_0, r_2) \tag{13}$$

$$C_1 = Tri_0(a_1, b_1, r_1) \oplus Tri_1(b_1, c_1, r_1) \oplus Tri_2(c_1, a_1, r_2) \tag{14}$$

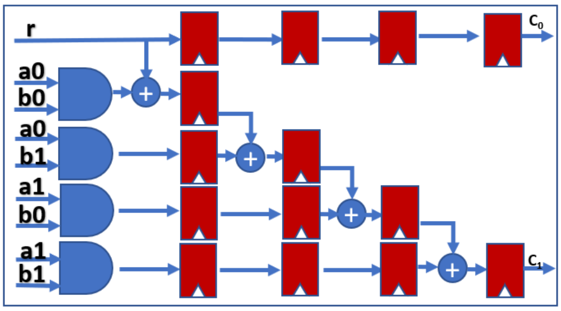


Figure 27: Trichina AND Gate

Figure 28 shows the 1-bit ripple-carry style security-masked adder with the Trichina AND gate used in the RCA.

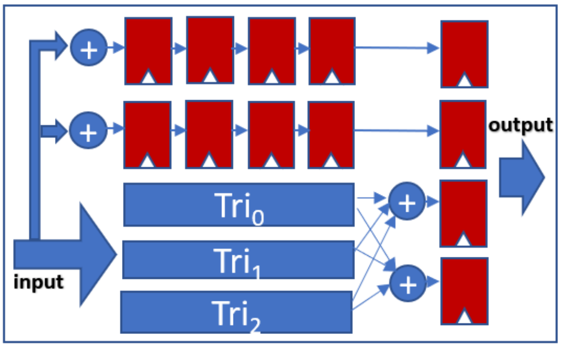


Figure 28: 1-bit Security-Masked Adder

For the final 1-bit security-masked adder, 12 more registers (marked in red) complement the Trichina AND gate to address the introduced delay in the Trichina AND gate. In total, for 1-bit security-masked adder, 54 $(14 * 3 + 12 = 54)$ registers are added. Generally, an N-bit ripple carry adder is formed by N 1-bit full adders, as shown in Figure 29. Therefore, for one N-bit security-masked adder, the total number of registers introduced to the design is $54 \cdot N$. Depending on the adder's size N and the number of summation resources instantiated in the BNN, a significant number of registers may be introduced that increase the susceptibility of the design to soft-errors.

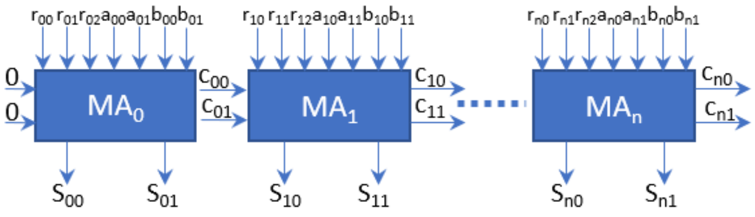

$r_{00}\ r_{01} r_{02} a_{00} a_{01} b_{00} b_{01}$
$r_{10}\ r_{11} r_{12} a_{10} a_{11} b_{10} b_{11}$
$r_{n0}\ r_{n1} r_{n2} a_{n0} a_{n1} b_{n0} b_{n1}$
0
0
MA_0
MA_1
MA_n
C_{00}
C_{01}
C_{10}
C_{11}
C_{n0}
C_{n1}
S_{00}
S_{01}
S_{10}
S_{11}
S_{n0}
S_{n1}

Figure 29: N-bit Masked Adder

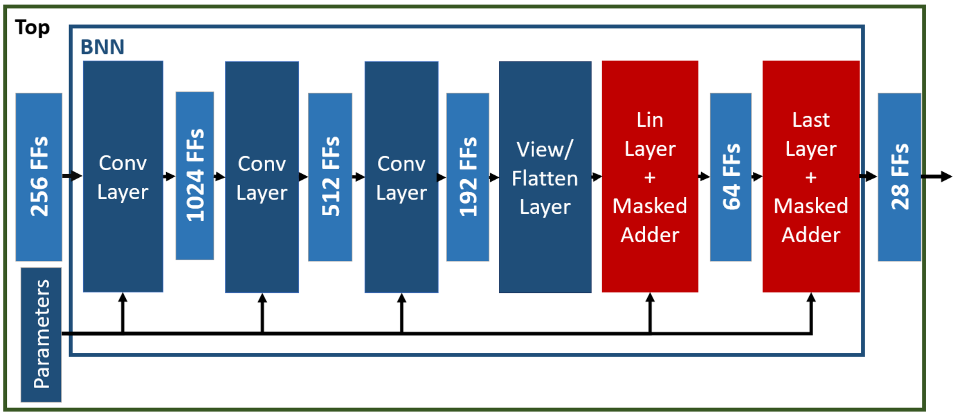


Figure 30: A Case-Study Security-Masked BNN Design

5.4.2 Security-Masked BNN-Based Case Study Design

In order to study how the weighted summation masking technique influences the soft-error reliability of the BNN hardware implementation, we apply the above masking technique on a case-study ultra-low power near-sensor binarized neural network implementation proposed in [174]. There are six layers in total which include three convolutional layers, one flattening layer and two fully connected layers. The DPA countermeasure target masking technique is initially applied to the fully connected layers [152] and the last two layers of the BNN hardware implementation influence the output the most, therefore we apply the masking technique to the last two fully connected layers in the BNN. We replace the weighted summation operation (i.e., popcount) with the N-bit security-masked adder explained in section 5.4.1: 8-bit security-masked adder and 7-bit security-masked adder for layers 5 and 6 separately. Due to the number of summation resources, the security-masked design introduces around 5 million flip-flops. The final design is shown in the Figure 30 and the security-masked layers are marked in red.

5.4.3 Reliability Assessment Setup

Replacing an original adder with the security-masked adder introduces many additional flip-flops to the circuit. Each additional flip-flop, in turn, increases the probability of SEU faults. However, soft-error effects can be masked as discussed in Section 5.3. To evaluate the actual reliability jeopardy by the added flip-flops and to compare the resulting failure rate of the original BNN with the security-masked BNN design, the de-rating factors have to be computed.

The case-study BNN implementation [174] provides a functional testbench which applies 100 different input data samples (images) to the network and the related outputs are categorised into 4 classes. For the experiment, the testbench is modified in the way that all bit-wise changes of the BNN outputs as well as the predicted class (i.e., the functional result) are monitored and recorded for each applied input. This simulation serves as the golden run.

The faults, i.e., inverted values of the target flip-flops, are injected at a random target and at a random clock cycle during the active workload of the simulation. Thus, each simulation run is independent and several runs can be performed in parallel. Since the simulation time of the original (i.e., non-masked) BNN design is rather short it is possible to run a complete fault injection campaign. This means that for each target flip-flop enough fault injection simulations are performed to cover every clock cycle during the active stimuli of the testbench. For the modified security-masked case-study BNN design (i.e., with the implemented DPA countermeasures) the number of targets and the simulation time are significantly higher and therefore, the random fault injection sampling approach has to be used.

For each simulation run, the output of the circuit is monitored and any difference to the golden run is traced as an output failure. Additionally, the predicted class is compared to the golden simulation and the predicted class altered a functional failure is traced.

To obtain the de-rating factors for the output or functional failure, the number of observed failures is divided by the total number of injected faults. The failure rate is then calculated using Equation (8). Since the FIT rate of the flip-flops depends on the applied nano-scale implementation technology and the analysis in this work is performed on a higher level, for both designs, a normalized FIT rate of $\text{FIT}_{\text{FF, SEU}} = 1$ is assumed for each flip-flop. Equation (8) is simplified to multiply the number of flip-flops with the de-rating factor to obtain the final failure rate SER_{SEU} of the design.

5.5 Experimental Results

5.5.1 LDR Analysis of the Masked Full Adder

In order to analyse the critical parts of the BNN, an evaluation of the logical de-rating is performed on the (security-masked) ripple carry adder. The analysis is performed with the SoCFIT tool by IROC, as described in section 5.3.2. Different sizes (N) of the adder are considered and the results for the maximum and average LDR are shown in Table 6.

Table 6: LDR Analysis of the (Masked) Full Adder

	Full Adder		Masked Full Adder	
Size N (bits)	LDR (avg)	LDR (max)	LDR (avg)	LDR (max)
4	0.61	1.0	0.98	1.0
8	0.57	1.0	0.99	1.0
16	0.55	1.0	0.99	1.0

The maximum LDR is 1.0 for both adders and unaffected by the adder size N. This means, considering the worst case every fault in one of the sequential cells of the adder is propagating to the output of the adder. The average LDR, however, is decreasing for the normal full adder and increasing for the masked full adder. Generally, the LDR for the full adder is lower than the LDR for the masked full adder. This means that a fault in any of the sequential cells of the masked full adder is more likely to propagate to the output than a fault in the conventional full adder.

The used LDR analysis approach is pessimistic because it shows an upper bound for the LDR of the cell, without using any information of the actual workload. Thus, LDR is the probability for a fault affecting an internal cell to propagate to the primary outputs of the circuit. It does not include any aspects related to the criticality of the fault. Additionally, the BNN consists of different stages which contain other functional blocks besides the adder. An exhaustive analysis should therefore, consider the adder in the full context. This is achieved by performing an exhaustive fault-injection simulation campaign of the full BNN/masked BNN design. This campaign determines the Functional De-Rating factors and is presented in the next section.

5.6 Fault-Injection Simulation Campaign

This section presents the results obtained from the performed fault injection simulation campaigns. As mentioned in the previous section, the simulation time of the masked BNN design is significantly higher and thus, a full fault injection campaign over the entire available test data was not feasible. A random sampling fault injection campaign was performed instead and in order to confirm that enough fault injection simulations have been sampled, several simulation campaigns with a varying number of fault injections were executed. Figure 31 shows the de-rating factors for the output and functional failure measured from these campaigns depending on the number of injected faults. It can be seen that that the de-rating factors are converging and the carried out fault injection simulations were sufficient.

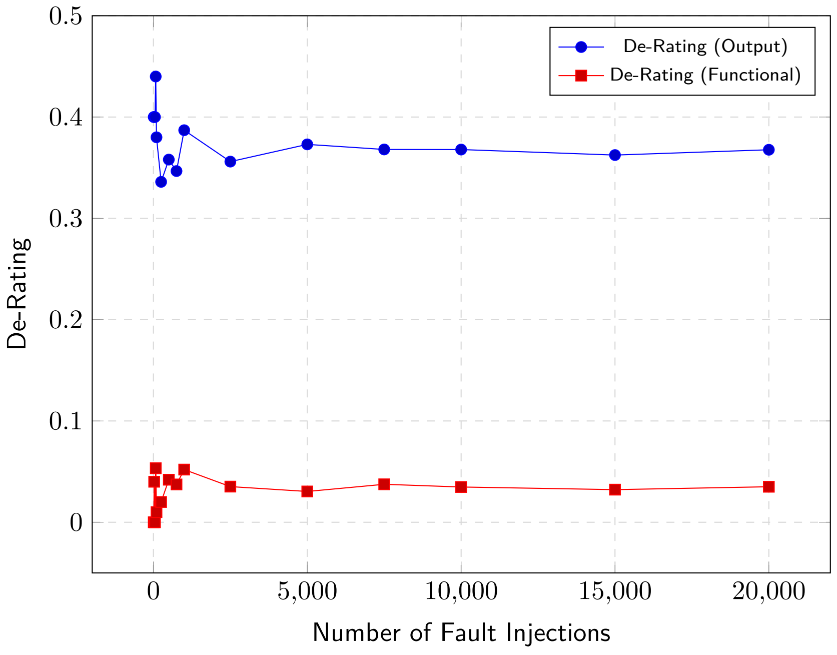


Figure 31: Convergence of the Masked BNN De-Rating Factors Values in the Random Sampling Fault-Injection Campaign

Table 7: Fault Injection Results

	BNN	Masked BNN
Flip-Flops	2076.00	5 348 912.00
Number of Injections	197 220.00	62 760.00
Output Failures	188 186.00	23 000.00
De-Rating (Output)	0.95	0.37
Failure Rate (Output)	1980.91	1 960 245.00
Functional Failures	48 948.00	2165.00
De-Rating (Functional)	0.25	0.03
Failure Rate (Functional)	515.24	184 518.71

All the simulation campaigns shown in Figure 31 run a different subset of fault injections (different target flip-flops with different input data) and therefore, can be accumulated to obtain the overall de-rating and failure rate of the masked case-study BNN design. The results are summarized in Table 7 together with the results of the full fault injection campaign of the original case-study BNN design.

Table 7 provides a comparison of the original BNN and the masked BNN designs with regard to their expected reliability. Although the output and functional de-rating are about $3\times$ and $10\times$ lower for the masked BNN, due to the considerable high amount of additional flip-flops, the resulting failure rate is about $1000\times$ higher for the output and $350\times$ higher for the functional failures. This proves that the studied masking techniques used in the literature for power side-channel countermeasures significantly increase vulnerability of the HW BNN inference engines to soft errors.

5.7 Chapter Conclusions

Hardware BNN inference engines are gaining their popularity in the computer engineering domain and stamp their inevitable presence in the security and reliability-critical applications. However, the efforts of security and reliability R&D communities happen to be fragmented to address the issues specific to their domains. A prominent example is a power side-channel countermeasure based on a set of masking techniques introduced to address a recently discovered security vulnerability. The approach overlooks potential issues for reliability and creates a significant vulnerability to radiation-induced single event effect faults, i.e., soft errors, in the field.

In this chapter, first, the study has presented an analysis for the soft-error reliability jeopardy by the DPA side-channel mitigation measures in hardware implementations of BNN inference engines. The analysis contains the specific reliability assessment, the fault injection simulation campaign and the analysis of the simulation results. This work reveals reliability issues, i.e., a steep increase of vulnerability to single-event effects, introduced by the security enhancement techniques. The results demonstrate that due to the considerable number of additional flip-flops established by the security countermeasure and their chained position, the bit-wise failure rate is about $1000\times$ higher for the BNN output and $350\times$ higher for the neural network functional failures.

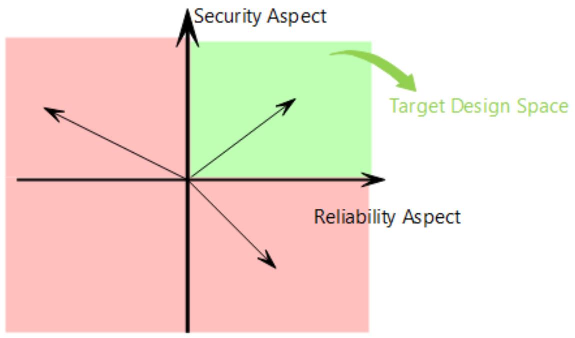


Figure 32: Security and Reliability Design Space Exploration

Second, this study also for the first time emphasizes the interdependency of the design space between reliability and security aspects, which is essential for the critical embedded system. As indicated in Figure 32, the vertical and horizontal axes represent design space targeting security and reliability requirements separately. The red area represents the situation that design modifications targeted at the improvement for one aspect have negative influence on the other aspect. While for the green area, it indicates design modification targeted at one aspect has positive influences on the other aspect, which is also the ideal case when dealing with multiple extra-functional aspects' requirements. In this research work, the final results have shown that the introduced secure enhancement modification to the BNN design has negative effects on its reliability which falls in the red area in Figure 32.

6 Conclusions

Recent prominent trends in advanced electronic systems have emphasized the importance of extra-functional aspects such as security, reliability and power. However, a systematic study for extra-functional aspects' verification from hardware perspective is still missing. This PhD thesis contributes to closing the gap and targets at extra-functional verification in hardware design with the focus on the reliability and security aspects.

The research work starts with a systematic study of the extra-functional aspects from a hardware design perspective. **Chapter 3** provides the **first systematic review** on the status of hardware extra-functional verification based on a comprehensive literature analysis. Based on the review, an up-to-date taxonomy of extra-functional aspects was proposed. It also identifies the research challenges for today's extra-functional aspects' verification. In particular, the analysis shows that the state-of-the-art methodologies and tools are rarely considering the interdependency of extra-functional aspects.

Based on the results of the study in **Chapter 3**, the security and reliability aspects were identified as the most widely addressed both in academia and industry, and selected for further study in **Chapter 4** and **Chapter 5** correspondingly.

- **Chapter 4** targets timing SCA related security aspects verification, which is conventionally analyzed after manufacturing and, therefore, results in significant costs and the increase of the time-to-market. Aiming to provide an early design stage verification of timing side-channel attack to reduce the design cost and time, it presents both formal and simulation-based design verification methods.

  The formal approach, presented in **4.2**, focuses on a crypto core, which contributes to secure communication in the electronic system. The verification method combines the information flow tracking with the formal tool JasperGold SPV to identify all the timing channels which lead to secure information (a secret key) leakage. The method is able to automatically and formally identify all the timing channels, and its feasibility was proved by a case study on RSA crypto core. The study includes algorithm analysis to determine the timing leakages part (e.g., various encryption/decryption times caused by different values of the secret keys) and additional design modification to assist the tool to find all the timing side channels. Additionally, a lightweight and effective mitigation technique to balance all the timing side channels was proposed.

  The other approach, i.e. simulation-based methods presented in **4.3**, targets the ECC decoder used in the fuzzy extractors for PUF devices. First, this work defines a novel attack model that combines fault injection techniques and timing side-channel analysis to exploit the PUF secrets. The study involves definition of a new attack model, the attack assumptions and the detailed attack procedure that combines the fault injection for the memory-based PUF device with timing-channel measurements for the ECC decoder used in the fuzzy extractor. Second, this work provides a simulation-based design verification method to verify the vulnerability of the design at early RTL design phase. The verification method contains a structural analysis step to screen the input parameters and a simulation-based analysis step to verify these parameters exhaustively.

  In general, the work presented in **Chapter 4** highlights the ability to verify the timing side-channel attack at the early design phase contrary to the vast majority of approaches available in the literature that do it after the manufacturing. This enables the early detection of related design bugs that still can be easily fixed during the early design phase, thus saving the design cost and reducing the time-to-market.

- **Chapter 5** concentrates on reliability study of security-enhanced BNN hardware inference engines. It is the first time, when such a study addresses the mutual influence between security and reliability aspects and explores the verification methodology that can measure the influences of the security enhancements on the design reliability quantitatively. First, the work analyzes the security mitigation techniques used for protecting against power side-channel vulnerability for BNN. One of such representative mitigation methods introduced a significant amount of registers which are acting as "magnets" for the soft-errors, and therefore creating a vulnerability. Then the research work has applied this mitigation method to the BNN hardware inference engine design. Accordingly, specific soft-error reliability assessments was created and then the related verification strategy was proposed. The final simulation results quantitatively proved that the security enhancement part of the circuit has a significant impact on the reliability of the design.

The **future work** for the research beyond this PhD thesis can fall into the vertical and horizontal directions. For the horizontal direction, it can continue the research work towards security and reliability aspect verification for BNN hardware inference engines, but in the reverse direction, i.e. studying how the reliability techniques affect the security of the network. For the vertical direction, the future work can target a holistic method to verify different extra-functional aspects at the same time by increasing the design abstraction levels or using model-checking based verification methods.

List of Figures

List of Tables

References

[1] M. Jenihhin, X. Lai, T. Ghasempouri, and J. Raik, "Towards multidimensional verification: Where functional meets non-functional," in *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–7, IEEE, 2018.

[2] H. Li, W. Zhang, S. Bhunia, and W. Wen, "Introduction to the special issue on new trends in nanoelectronic device, circuit, and architecture design, part 1," 2020.

[3] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *14th IEEE International Requirements Engineering Conference (RE'06)*, pp. 39–48, IEEE, 2006.

[4] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the internet of things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, 2014.

[5] D. Tang, C. He, Y. Li, H. Zang, C. Xiong, and J. Zhang, "Soft error reliability in advanced cmos technologies-trends and challenges," *Science China Technological Sciences*, vol. 57, no. 9, pp. 1846–1857, 2014.

[6] M. Glinz, "On non-functional requirements," in *15th IEEE international requirements engineering conference (RE 2007)*, pp. 21–26, IEEE, 2007.

[7] M. Younas, D. N. Jawawi, I. Ghani, and M. A. Shah, "Extraction of non-functional requirement using semantic similarity distance," *Neural Computing and Applications*, vol. 32, no. 11, pp. 7383–7397, 2020.

[8] F. Z. Hammani, "Survey of non-functional requirements modeling and verification of software product lines," in *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, pp. 1–6, IEEE, 2014.

[9] D. Mukhopadhyay and R. S. Chakraborty, *Hardware security: design, threats, and safeguards*. CRC Press, 2014.

[10] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[11] W. K. Lam, *Hardware design verification: simulation and formal method-based approaches*. Prentice Hall Professional Technical Reference, 2005.

[12] I. Sommerville, "Software engineering 9th edition," *ISBN-10*, vol. 137035152, p. 18, 2011.

[13] S. Bhunia and M. Tehranipoor, *Hardware security: a hands-on learning approach*. Morgan Kaufmann, 2018.

[14] I. Eusgeld, B. Fechner, F. Salfner, M. Walter, P. Limbourg, and L. Zhang, "Hardware reliability," in *Dependability metrics*, pp. 59–103, Springer, 2008.

[15] A. P. Shah and P. Girard, “Impact of aging on soft error susceptibility in cmos circuits,” in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 1–4, IEEE, 2020.

[16] X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, “Understanding multidimensional verification: Where functional meets non-functional,” *Microprocessors and Microsystems*, vol. 71, p. 102867, 2019.

[17] I. Verbauwhede, “Security adds an extra dimension to ic design: Future ic design must focus on security in addition to low power and energy,” *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, pp. 41–45, 2017.

[18] W. Chen, S. Ray, J. Bhadra, M. Abadir, and L.-C. Wang, “Challenges and trends in modern soc design verification,” *IEEE Design & Test*, vol. 34, no. 5, pp. 7–22, 2017.

[19] J. Bhadra, M. S. Abadir, L.-C. Wang, and S. Ray, “A survey of hybrid techniques for functional verification,” *IEEE Design & Test of Computers*, vol. 24, no. 02, pp. 112–122, 2007.

[20] M. L. Fair, C. R. Conklin, S. B. Swaney, P. J. Meaney, W. J. Clarke, L. C. Alves, I. N. Modi, F. Freier, W. Fischer, and N. E. Weber, “Reliability, availability, and serviceability (ras) of the ibm eserver z990,” *IBM Journal of Research and Development*, vol. 48, no. 3.4, pp. 519–534, 2004.

[21] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*, vol. 5. Springer Science & Business Media, 2012.

[22] P. Singh and A. K. Tripathi, “Exploring problems and solutions in estimating testing effort for non functional requirement,” *International Journal of Computers & Technology*, vol. 3, no. 2b, pp. 284–290, 2012.

[23] E. R. Poort, N. Martens, I. Van De Weerd, and H. Van Vliet, “How architects see non-functional requirements: Beware of modifiability,” in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 37–51, Springer, 2012.

[24] D. Ameller, C. Ayala, J. Cabot, and X. Franch, “How do software architects consider non-functional requirements: An exploratory study,” in *2012 20th IEEE international requirements engineering conference (RE)*, pp. 41–50, IEEE, 2012.

[25] L. Motus, “Analytical study of quantitative timing properties of software,” in *Fifth Euromicro Workshop on Real-Time Systems*, pp. 218–223, IEEE, 1993.

[26] A. Pfitzmann and M. Hansen, “Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology,” *Version v0*, vol. 31, p. 15, 2008.

[27] H. M. Wassel, Y. Gao, J. K. Oberg, T. Huffmire, R. Kastner, F. T. Chong, and T. Sherwood, “Networks on chip with provable security properties,” *IEEE Micro*, vol. 34, no. 3, pp. 57–68, 2014.

[28] L.-W. Kim and J. D. Villasenor, "Dynamic function verification for system on chip security against hardware-based attacks," *IEEE transactions on reliability*, vol. 64, no. 4, pp. 1229–1242, 2015.

[29] A. Nahiyan, M. Sadi, R. Vittal, G. Contreras, D. Forte, and M. Tehranipoor, "Hardware trojan detection through information flow security verification," in *2017 IEEE International Test Conference (ITC)*, pp. 1–10, IEEE, 2017.

[30] M. Yoshimura, T. Bouyashiki, and T. Hosokawa, "A hardware trojan circuit detection method using activation sequence generations," in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 221–222, IEEE, 2017.

[31] F. K. Lodhi, S. R. Hasan, O. Hasan, and F. Awwad, "Formal analysis of macro synchronous micro asychronous pipeline for hardware trojan detection," in *2015 Nordic Circuits and Systems Conference (NORCAS): NORCHIP & International Symposium on System-on-Chip (SoC)*, pp. 1–4, IEEE, 2015.

[32] J. Hansson, B. Lewis, J. Hugues, L. Wrage, P. Feiler, and J. Morley, "Model-based verification of security and non-functional behavior using aadl," *IEEE Security & Privacy*, pp. 1–1, 2009.

[33] T. Boraten, D. DiTomaso, and A. K. Kodi, "Secure model checkers for network-on-chip (noc) architectures," in *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 45–50, IEEE, 2016.

[34] M. A. Kochte, M. Sauer, L. R. Gomez, P. Raiola, B. Becker, and H.-J. Wunderlich, "Specification and verification of security in reconfigurable scan networks," in *2017 22nd IEEE European Test Symposium (ETS)*, pp. 1–6, IEEE, 2017.

[35] W. Hu, A. Becker, A. Ardeshiricham, Y. Tai, P. Ienne, D. Mu, and R. Kastner, "Imprecise security: quality and complexity tradeoffs for hardware information flow tracking," in *Proceedings of the 35th International Conference on Computer-Aided Design*, pp. 1–8, 2016.

[36] Z. Hanna and V. Purri, "Verifying security aspects of soc designs with jasper app," *(white paper), Jasper Design Automation (Cadence)*, 2013.

[37] M. A. Kochte, R. Baranowski, M. Sauer, B. Becker, and H.-J. Wunderlich, "Formal verification of secure reconfigurable scan network infrastructure," in *2016 21th IEEE European Test Symposium (ETS)*, pp. 1–6, IEEE, 2016.

[38] M. Rocchetto and N. O. Tippenhauer, "Towards formal security analysis of industrial control systems," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 114–126, 2017.

[39] J. Sepúlveda, D. Aboul-Hassan, G. Sigl, B. Becker, and M. Sauer, "Towards the formal verification of security properties of a network-on-chip router," in *2018 IEEE 23rd European Test Symposium (ETS)*, pp. 1–6, IEEE, 2018.

[40] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *Journal of Cryptographic Engineering*, vol. 8, no. 1, pp. 1–27, 2018.

[41] Y. Lyu and P. Mishra, "A survey of side-channel attacks on caches and countermeasures," *Journal of Hardware and Systems Security*, vol. 2, no. 1, pp. 33–50, 2018.

[42] S. Deng, W. Xiong, and J. Szefer, "Cache timing side-channel vulnerability checking with computation tree logic," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, pp. 1–8, 2018.

[43] T. Zhang and R. B. Lee, "New models of cache architectures characterizing information leakage from cache side channels," in *Proceedings of the 30th annual computer security applications conference*, pp. 96–105, 2014.

[44] P. Cañones, B. Köpf, and J. Reineke, "Security analysis of cache replacement policies," in *International Conference on Principles of Security and Trust*, pp. 189–209, Springer, 2017.

[45] J. Vain, A. Kaur, L. Tsiopoulos, J. Raik, and M. Jenihhin, "Multi-view modeling for mpsoc design aspects," in *2018 16th Biennial Baltic Electronics Conference (BEC)*, pp. 1–6, IEEE, 2018.

[46] "Jaspergold security path verification app, cadence." `http://www.cadence.com`.

[47] F. S. Goncalves, D. Pereira, E. Tovar, and L. B. Becker, "Formal verification of aadl models using uppaal," in *2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 117–124, IEEE, 2017.

[48] D. Burlyaev and P. Fradet, "Formal verification of automatic circuit transformations for fault-tolerance," in *2015 Formal Methods in Computer-Aided Design (FMCAD)*, pp. 41–48, IEEE, 2015.

[49] B. W. Thompto and B. Hoppe, "Verification for fault tolerance of the ibm system z microprocessor," in *Design Automation Conference*, pp. 525–530, IEEE, 2010.

[50] S. Kan, M. Lam, T. Porter, and J. Dworak, "A case study: pre-silicon soc ras validation for noc server processor," in *2016 17th International Workshop on Microprocessor and SOC Test and Verification (MTV)*, pp. 19–24, IEEE, 2016.

[51] S. Vinco, M. Lora, E. Macii, and M. Poncino, "Ip-xact for smart systems design: extensions for the integration of functional and extra-functional models," in *2016 Forum on Specification and Design Languages (FDL)*, pp. 1–8, IEEE, 2016.

[52] S. Vinco, Y. Chen, F. Fummi, E. Macii, and M. Poncino, "A layered methodology for the simulation of extra-functional properties in smart systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1702–1715, 2017.

[53] G. Aleksandrowicz, E. Arbel, R. Bloem, T. D. ter Braak, S. Devadze, G. Fey, M. Jenihhin, A. Jutman, H. G. Kerkhoff, R. Könighofer, *et al.*, "Designing reliable cyber-physical systems," in *Languages, Design Methods, and Tools for Electronic System Design*, pp. 15–38, Springer, 2018.

[54] E. Arbel, S. Koyfman, P. Kudva, and S. Moran, "Automated detection and verification of parity-protected memory elements," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2014.

[55] M. Maniatakos and Y. Makris, "Workload-driven selective hardening of control state elements in modern microprocessors," in *2010 28th VLSI test symposium (VTS)*, pp. 159–164, IEEE, 2010.

[56] S. Ganapathy, R. Canal, D. Alexandrescu, E. Costenaro, A. González, and A. Rubio, "Informer: An integrated framework for early-stage memory robustness analysis," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–4, IEEE, 2014.

[57] S. Avramenko, S. P. Azad, S. Esposito, B. Niazmand, M. Violante, J. Raik, and M. Jenihhin, "Qosinnoc: analysis of qos-aware noc architectures for mixed-criticality applications," in *2018 IEEE 21st International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pp. 67–72, IEEE, 2018.

[58] S. Avramenko, S. P. Azad, B. Niazmand, M. Violante, J. Raik, and M. Jenihhin, "Upgrading qosinnoc: efficient routing for mixed-criticality applications and power analysis," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 207–212, IEEE, 2018.

[59] S. Rubini, F. Singhoff, and J. Hugues, "Modeling and verification of memory architectures with aadl and real," in *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*, pp. 338–343, IEEE, 2011.

[60] H. Wang, X. Zhou, Y. Dong, and L. Tang, "A hierarchical verification procedure of timed petri-net model for real-time embedded systems," in *2010 2nd International Conference on Information Engineering and Computer Science*, pp. 1–4, IEEE, 2010.

[61] H. Wang, X. Zhou, Y. Dong, and L. Tang, "Timing properties analysis of real-time embedded systems with aadl model using model checking," in *2010 IEEE International Conference on Progress in Informatics and Computing*, vol. 2, pp. 1019–1023, IEEE, 2010.

[62] A. Rafiev, F. Xia, A. Iliasov, A. Romanovsky, and A. Yakovlev, "Selective abstraction for estimating extra-functional properties in networks-on-chips using archon framework," in *2017 17th International Conference on Application of Concurrency to System Design (ACSD)*, pp. 80–85, IEEE, 2017.

[63] D. Lorenz, P. A. Hartmann, K. Grüttner, and W. Nebel, "Non-invasive power simulation at system-level with systemc," in *International Workshop on Power and Timing Modeling, Optimization and Simulation*, pp. 21–31, Springer, 2012.

[64] S. Orcioni, M. Giammarini, C. Scavongelli, G. B. Vece, and M. Conti, "Energy estimation in systemc with powersim," *Integration*, vol. 55, pp. 118–128, 2016.

[65] E.-Y. Kang, D. Mu, L. Huang, and Q. Lan, "Verification and validation of a cyber-physical system in the automotive domain," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 326–333, IEEE, 2017.

[66] A. Danese, G. Pravadelli, and I. Zandona, "Automatic generation of power state machines through dynamic mining of temporal assertions," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 606–611, IEEE, 2016.

[67] J. Zimmermann, S. Stattelmann, A. Viehl, O. Bringmann, and W. Rosenstiel, “Model-driven virtual prototyping for real-time simulation of distributed embedded systems,” in *7th IEEE International Symposium on Industrial Embedded Systems (SIES’12)*, pp. 201–210, IEEE, 2012.

[68] R. Görgen, K. Grüttner, F. Herrera, P. Penil, J. Medina, E. Villar, G. Palermo, W. Fornaciari, C. Brandolese, D. Gadioli, *et al.*, “Contrex: Design of embedded mixed-criticality control systems under consideration of extra-functional properties,” in *2016 Euromicro Conference on Digital System Design (DSD)*, pp. 286–293, IEEE, 2016.

[69] A. Ruan, Y. Liao, P. Li, W. Li, and W. Li, “Throughput estimation for modelsim simulator tool based hw/sw co-verification system,” in *2009 International Conference on Communications, Circuits and Systems*, pp. 1014–1018, IEEE, 2009.

[70] M. Khamis, S. El-Ashry, A. Shalaby, M. AbdElsalam, and M. W. El-Kharashi, “A configurable risc-v for noc-based mpsocs: a framework for hardware emulation,” in *2018 11th International Workshop on Network on Chip Architectures (NoCArc)*, pp. 1–6, IEEE, 2018.

[71] “Jaspergold connectivity verification app, cadence.” `http://www.cadence.com`.

[72] M. Elver and V. Nagarajan, “Mcversi: A test generation framework for fast memory consistency verification in simulation,” in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 618–630, IEEE, 2016.

[73] S.-C. Fang, C.-C. Weng, C.-K. Tseng, C.-W. Hsu, J.-L. Liao, S.-Y. Huang, C.-L. Lung, and D.-M. Kwai, “Soc power analysis framework and its application to power-thermal co-simulation,” in *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test*, pp. 1–4, IEEE, 2011.

[74] G. B. Vece, M. Conti, and S. Orcioni, “Transaction-level power analysis of vlsi digital systems,” *Integration*, vol. 50, pp. 116–126, 2015.

[75] M. Giammarini, M. Conti, and S. Orcioni, “System-level energy estimation with powersim,” in *2011 18th IEEE International Conference on Electronics, Circuits, and Systems*, pp. 723–726, IEEE, 2011.

[76] S. Mukherjee, *Architecture design for soft errors*. Morgan Kaufmann, 2011.

[77] M. Jenihhin, G. Squillero, T. S. Copetti, V. Tihhomirov, S. Kostin, M. Gaudesi, F. Vargas, J. Raik, M. S. Reorda, L. B. Poehls, *et al.*, “Identification and rejuvenation of nbti-critical logic paths in nanoscale circuits,” *Journal of Electronic Testing*, vol. 32, no. 3, pp. 273–289, 2016.

[78] A. Savino, S. Di Carlo, A. Vallero, G. Politano, D. Gizopoulos, and A. Evans, “Riif-2: Toward the next generation reliability information interchange format,” in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 173–178, IEEE, 2016.

[79] “Bell labs, verifying multi-threaded software with spin.” `http://spinroot.com`. accessed December 2021.

[80] "Smt steering committee, the international satisfiability modulo theories (smt) competition.." `http://www.smtcomp.org`. accessed December 2021.

[81] "Carnegie mellon university, architecture analysis and design language.." `http://www.aadl.info/aadllcurrentsite`. accessed December 2021.

[82] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[83] S. K. Roy, "Top level soc interconnectivity verification using formal techniques," in *2007 Eighth International Workshop on Microprocessor Test and Verification*, pp. 63–70, IEEE, 2007.

[84] "IEEE Standard for design and verification of Low-Power, energy-aware electronic systems,," *ANSI/IEEE 1801-2015*, March 2016.

[85] "Si2 Common Power Format, v2.1, Silicon Integration Initiative," 2014.

[86] R. Ubar, J. Raik, A. Jutman, and M. Jenihhin, "Diagnostic modeling of digital systems with multi-level decision diagrams," in *Geographic Information Systems: Concepts, Methodologies, Tools, and Applications*, pp. 407–433, IGI Global, 2013.

[87] P. Khondkar, *Low-Power Design and Power-Aware Verification*. Springer, 2018.

[88] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, *et al.*, "Machine learning for electronic design automation: A survey," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 26, no. 5, pp. 1–46, 2021.

[89] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[90] B. Yu, D. Z. Pan, T. Matsunawa, and X. Zeng, "Machine learning and pattern matching in physical design," in *The 20th Asia and South Pacific Design Automation Conference*, pp. 286–293, IEEE, 2015.

[91] B. Li and P. D. Franzon, "Machine learning in physical design," in *2016 IEEE 25th conference on electrical performance of electronic packaging and systems (EPEPS)*, pp. 147–150, IEEE, 2016.

[92] L. Bai and L. Chen, "Machine-learning-based early-stage timing prediction in soc physical design," in *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pp. 1–3, IEEE, 2018.

[93] S. Dai, Y. Zhou, H. Zhang, E. Ustun, E. F. Young, and Z. Zhang, "Fast and accurate estimation of quality of results in high-level synthesis with machine learning," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 129–132, IEEE, 2018.

[94] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware trojans classification for gate-level netlists based on machine learning," in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 203–206, IEEE, 2016.

[95] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware trojans classification for gate-level netlists using multi-layer neural networks," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 227–232, IEEE, 2017.

[96] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, IEEE, 2017.

[97] A. Piziali, *Functional verification coverage measurement and analysis*. Springer Science & Business Media, 2007.

[98] V. Guarnieri, N. Bombieri, G. Pravadelli, F. Fummi, H. Hantson, J. Raik, M. Jenihhin, and R. Ubar, "Mutation analysis for systemc designs at tlm," in *2011 12th Latin American Test Workshop (LATW)*, pp. 1–6, IEEE, 2011.

[99] S. Ullah, M. Iqbal, and A. M. Khan, "A survey on issues in non-functional requirements elicitation," in *International Conference on Computer Networks and Information Technology*, pp. 333–340, IEEE, 2011.

[100] L. M. Cysneiros and E. Yu, "Non-functional requirements elicitation," in *Perspectives on software requirements*, pp. 115–138, Springer, 2004.

[101] "S.-p. azad, b. niazmand, k. janson, j. raik, github bonfire project (2017." `Bonfireprojectwebsite:https://github.com/Project-Bonfire/[Online]`. accessed December 2021.

[102] S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, *et al.*, "From online fault detection to fault management in network-on-chips: A ground-up approach," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pp. 48–53, IEEE, 2017.

[103] A. S. Alkalbani, T. Mantoro, and A. O. M. Tap, "Comparison between rsa hardware and software implementation for wsns security schemes," in *Proceeding of the 3rd International Conference on Information and Communication Technology for the Moslem World (ICT4M) 2010*, pp. E84–E89, IEEE, 2010.

[104] S. Upadhyay, "Attack on rsa cryptosystem," 2011.

[105] D. Boneh *et al.*, "Twenty years of attacks on the rsa cryptosystem," *Notices of the AMS*, vol. 46, no. 2, pp. 203–213, 1999.

[106] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, no. 5, pp. 236–243, 1976.

[107] J. Ming, D. Wu, G. Xiao, J. Wang, and P. Liu, "Taintpipe: Pipelined symbolic taint analysis," in *24th USENIX Security Symposium (USENIX Security 15)*, (Washington, D.C.), pp. 65–80, USENIX Association, 2015.

[108] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of AES," in *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, pp. 1–20, 2006.

[109] A. Haeberlen, B. C. Pierce, and A. Narayan, "Differential privacy under fire," in *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*, 2011.

[110] B. B. Brumley and N. Tuveri, "Remote timing attacks are still practical," in *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, pp. 355–371, 2011.

[111] D. Zhang, A. Askarov, and A. C. Myers, "Predictive mitigation of timing channels in interactive systems," in *In Proc. 18th ACM Conf. Computer and Communications Security (CCS*, pp. 563–574, 2011.

[112] F. Koeune and F.-X. Standaert, "Foundations of security analysis and design iii," ch. A Tutorial on Physical Security and Side-channel Attacks, pp. 78–108, Berlin, Heidelberg: Springer-Verlag, 2005.

[113] B. Coppens, I. Verbauwhede, K. D. Bosschere, and B. D. Sutter, "Practical mitigations for timing-based side-channel attacks on modern x86 processors," in *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*, pp. 45–60, 2009.

[114] K. Tiri and I. Verbauwhede, "A vlsi design flow for secure side-channel attack resistant ics," in *Design, Automation and Test in Europe*, pp. 58–63 Vol. 3, March 2005.

[115] I. Verbauwhede and P. Schaumont, "Design methods for security and trust," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '07, (San Jose, CA, USA), pp. 672–677, EDA Consortium, 2007.

[116] F. Menichelli, R. Menicocci, M. Olivieri, and A. Trifiletti, "High-level side-channel attack modeling and simulation for security-critical systems on chips," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, pp. 164–176, July 2008.

[117] A. Ardeshiricham, W. Hu, J. Marxen, and R. Kastner, "Register transfer level information flow tracking for provably secure hardware design," in *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '17, (3001 Leuven, Belgium, Belgium), pp. 1695–1700, European Design and Automation Association, 2017.

[118] M. Bidmeshki and Y. Makris, "Toward automatic proof generation for information flow policies in third-party hardware ip," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, vol. 00, pp. 163–168, May 2015.

[119] D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers, "A hardware design language for timing-sensitive information-flow security," *SIGPLAN Not.*, vol. 50, pp. 503–516, Mar. 2015.

[120] S. Deng, W. Xiong, and J. Szefer, "Cache timing side-channel vulnerability checking with computation tree logic," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '18, (New York, NY, USA), pp. 2:1–2:8, ACM, 2018.

[121] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the aes s-box," in *Fast Software Encryption* (H. Gilbert and H. Handschuh, eds.), (Berlin, Heidelberg), pp. 413–423, Springer Berlin Heidelberg, 2005.

[122] L. S. Indrusiak, J. Harbin, and M. J. Sepúlveda, "Side-channel attack resilience through route randomisation in secure real-time networks-on-chip," *CoRR*, vol. abs/1607.03450, 2016.

[123] Z. Jiang, S. Dai, G. E. Suh, and Z. Zhang, "High-level synthesis with timing-sensitive information flow enforcement," in *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18, (New York, NY, USA), pp. 88:1–88:8, ACM, 2018.

[124] S. Peter and T. Givargis, "Towards a timing attack aware high-level synthesis of integrated circuits," in *34th IEEE International Conference on Computer Design, ICCD 2016, Scottsdale, AZ, USA, October 2-5, 2016*, pp. 452–455, 2016.

[125] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*, pp. 3–37, Springer, 2010.

[126] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for sram pufs," in *2009 IEEE international symposium on information theory*.

[127] A. R. Korenda, F. Afghah, B. Cambou, and C. Philabaum, "A proof of concept sram-based physically unclonable function (puf) key generation mechanism for iot devices," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–8, 2019.

[128] L. Tebelmann, M. Pehl, and G. Sigl, "Em side-channel analysis of bch-based error correction for puf-based key generation," in *Proceedings of the 2017 Workshop on Attacks and Solutions in Hardware Security*.

[129] D. Karakoyunlu and B. Sunar, "Differential template attacks on puf enabled cryptographic devices," in *2010 IEEE International Workshop on Information Forensics and Security*, pp. 1–6, IEEE, 2010.

[130] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*, Springer, 2004.

[131] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-channel analysis of pufs and fuzzy extractors," in *International Conference on Trust and Trustworthy Computing*, pp. 33–47, Springer, 2011.

[132] G. T. Becker, "Robust fuzzy extractors and helper data manipulation attacks revisited: Theory vs practice," 2017.

[133] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for puf-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2014.

[134] M. Riley and I. Richardson, "An introduction to reed-solomon codes: principles, architecture and implementation," 2003.

[135] C. Roscian, A. Sarafianos, J.-M. Dutertre, and A. Tria, "Fault model analysis of laser-induced faults in sram memory cells," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 89–98, IEEE, 2013.

[136] Y. Gao, Y. Su, W. Yang, S. Chen, S. Nepal, and D. C. Ranasinghe, "Building secure sram puf key generators on resource constrained devices," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*.

[137] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *International workshop on cryptographic hardware and embedded systems*, pp. 2–12, Springer, 2002.

[138] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "A side channel attack countermeasure using system-on-chip power profile scrambling," in *2011 IEEE 17th International On-Line Testing Symposium*, pp. 222–227, IEEE, 2011.

[139] X. Lai, M. Jenihhin, J. Raik, and K. Paul, "Pascal: Timing sca resistant design and verification flow," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 239–242, IEEE, 2019.

[140] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput bch error correction vlsi design for multi-level cell nand flash memories," in *2006 IEEE Workshop on Signal Processing Systems Design and Implementation*.

[141] H.-C. Chang and C. B. Shung, "New serial architecture for the berlekamp-massey algorithm," *IEEE transactions on communications*, 1999.

[142] S. Lee and H. Lee, "A high-speed pipelined degree-computationless modified euclidean algorithm architecture for reed-solomon decoders," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 91, no. 3, pp. 830–835, 2008.

[143] "Verilog based bch encoder / decoder." `https://github.com/russdill/bch_verilog`.

[144] "Freecores reed-solomon codec generator." `https://github.com/freecores/reed_solomon_codec_generator`.

[145] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Pro. of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[146] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 159–172, 2018.

[147] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[148] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*, pp. 525–542, Springer, 2016.

[149] A. Dubey, R. Cammarota, and A. Aysu, "Maskednet: The first hardware inference engine aiming power side-channel protection," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 197–208, IEEE, 2020.

[150] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), (Berlin, Heidelberg), pp. 388–397, Springer Berlin Heidelberg, 1999.

[151] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*, pp. 16–29, Springer, 2004.

[152] A. Dubey, R. Cammarota, and A. Aysu, "Bomanet: boolean masking of an entire neural network," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2020.

[153] M. A. Hanif, F. Khalid, R. V. W. Putra, S. Rehman, and M. Shafique, "Robust machine learning systems: Reliability and security for deep neural networks," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pp. 257–260, IEEE, 2018.

[154] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

[155] R. N. Reith, T. Schneider, and O. Tkachenko, "Efficiently stealing your machine learning models," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, WPES'19, (New York, NY, USA), p. 198–210, Association for Computing Machinery, 2019.

[156] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, (Santa Clara, CA), pp. 515–532, USENIX Association, Aug. 2019.

[157] H. Yu, H. Ma, K. Yang, Y. Zhao, and Y. Jin, "Deepem: Deep neural networks model recovery through em side-channel information leakage," *2020 IEEE Int. Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 209–218, 2020.

[158] E. Trichina, "Combinational logic design for aes subbyte transformation on masked data.," *IACR Cryptol. EPrint Arch.*, vol. 2003, p. 236, 2003.

[159] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, 2017.

[160] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, Boston, MA, 2011.

[161] D. Alexandrescu and E. Costenaro, "Towards optimized functional evaluation of see-induced failures in complex designs," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, pp. 182–187, 2012.

[162] S. Mittal, "A survey on modeling and improving reliability of dnn algorithms and accelerators," *Journal of Systems Architecture*, vol. 104, p. 101689, 2020.

[163] G. Abich, J. Gava, R. Reis, and L. Ost, "Soft error reliability assessment of neural networks on resource-constrained iot devices," in *2020 27th IEEE Int. Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4, IEEE, 2020.

[164] Y. Ibrahim, H. Wang, J. Liu, J. Wei, L. Chen, P. Rech, K. Adam, and G. Guo, "Soft errors in dnn accelerators: A comprehensive review," *Microelectronics Reliability*, vol. 115, p. 113969, 2020.

[165] Y. He, P. Balaprakash, and Y. Li, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 270–281, IEEE, 2020.

[166] W. Li, G. Ge, K. Guo, X. Chen, Q. Wei, Z. Gao, Y. Wang, and H. Yang, "Soft error mitigation for deep convolution neural network on fpga accelerators," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–5, 2020.

[167] B. Salami, O. S. Unsal, and A. C. Kestelman, "On the resilience of rtl nn accelerators: Fault characterization and mitigation," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 322–329, 2018.

[168] A. Ruospo, A. Bosio, A. Ianne, and E. Sanchez, "Evaluating convolutional neural networks reliability depending on their data representation," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pp. 672–679, 2020.

[169] M. A. Neggaz, I. Alouani, S. Niar, and F. Kurdahi, "Are cnns reliable enough for critical applications? an exploratory study," *IEEE Design Test*, vol. 37, no. 2, pp. 76–83, 2020.

[170] L.-H. Hoang, M. A. Hanif, and M. Shafique, "Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *Proc. Design, Automation and Test in Europe*, DATE '20, (San Jose, CA, USA), p. 1241–1246, EDA Consortium, 2020.

[171] E. Trichina, T. Korkishko, and K. H. Lee, "Small size, low power, side channel-immune aes coprocessor: design and synthesis results," in *International Conference on Advanced Encryption Standard*, pp. 113–127, Springer, 2004.

[172] O. Reparaz, R. de Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "Additively homomorphic ring-lwe masking," in *Post-Quantum Cryptography*, pp. 233–244, Springer, 2016.

[173] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked aes hardware implementations," in *Cryptographic Hardware and Embedded Systems – CHES 2005* (J. R. Rao and B. Sunar, eds.), (Berlin, Heidelberg), pp. 157–171, Springer Berlin Heidelberg, 2005.

[174] M. Rusci, L. Cavigelli, and L. Benini, “Design automation for binarized neural networks: A quantum leap opportunity?,” *CoRR*, vol. abs/1712.01743, 2017.

[175] X. Lai, M. Jenihhin, G. Selimis, S. Goossens, R. Maes, and K. Paul, “Early rtl analysis for sca vulnerability in fuzzy extractors of memory-based puf enabled devices,” in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pp. 16–21, IEEE, 2020.

[176] X. Lai, T. Lange, A. Balakrishnan, D. Alexandrescu, and M. Jenihhin, “On antagonism between side-channel security and soft-error reliability in bnn inference engines,” in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2021.

Acknowledgements

Finally, I reach this chapter with lots of gratitude. The accomplishment of this PhD thesis has received lots of supports from many people, and I would like to express my sincere thanks to all the people I have worked and cooperated with.

In particular, I would like to express my heartfelt appreciation to my supervisor, Profesor Maksim Jenihhin, for his tremendous supports, mentoring, and instructive advice throughout my PhD studies. It is a great and valuable experience to work with you. I would also thank Professor Jaan Raik for his collaboration and timely support when it was needed.

Special thanks to the Head of Department of Computer System, Dr. Margus Kruus, for the support. And I would also express my thanks to my colleagues at the Department of Computer Systems and researchers and supervisors from the RESCUE project.

Last but not least, sincere thanks to my parents and my siblings, who have and will always be there to support me.

I would like to acknowledge the organization that has supported my PhD studies: RESCUE European Training Network (ETN) project that received funding from European Union's Horizon 2020 research and innovation programme under the Marie Sklodowaska-Curie grant agreement No 722325.

Abstract
Approaches to Extra-Functional Verification of Security and Reliability Aspects in Hardware Designs

Several recent trends can be observed in advanced electronics. First, the technology shrinking has enabled sophisticated functions to be integrated into small-size chips. Such high-density integrated circuits accommodate billions of transistors, making the design sensitive to the environment and thus error-prone. This fact introduces the design and verification challenges for functional and extra-functional aspects. Second, along with the widespread of the Internet-of-Things and critical systems, the reliability, security, and power extra-functional aspects start playing essential roles to ensure design's functional correctness in harsh environments. Third, in order to cope up with very large and complex designs, Intellectual Property (IP) reuse methodology is introduced and widely applied to increase the design efficiency. This brings the front hardware security issues, and in particular, IP protection against hardware trojans potentially sneaking in with an external IP.

In order to align with the strict requirements for extra-functional aspects, the related design verification techniques have to evolve at the same pace. Unfortunately, the extra-functional verification techniques are still immature, and here the hardware design discipline is still to catch up with the developments in software engineering. Normally extra-functional aspects are composed of the measurement of the sequences of functional behaviors. For some extra-functional aspects such as reliability and security, the highly abstract extra-functional requirements need enormous analysis to extract the functional sequence with an in-depth understanding of the design. Extra-functional features, like side-channel leaks related to the security aspect, are usually verified only after the manufacturing step, thus causing significant design cost overheads. For verification of multiple extra-functional aspects, the problem becomes even more sophisticated, and the complexity often grows exponentially. Therefore, the objectives of the thesis are to have a comprehensive study of extra-functional aspects and address limitations of the existing hardware design verification methodology, with the focus on research of the most critical extra-functional aspects, i.e. security and reliability. The contributions of the performed research are summarized as follows.

- **A comprehensive understanding of the extra-functional aspects and features of the related design verification methodologies.** The research work is based on the literature review of the state-of-the-art for extra-functional aspects verification and proposes a taxonomy of extra-functional aspects. The research points out challenges for the current hardware design verification and proposes an initial method involving machine learning techniques to cope with the extra-functional features analysis.

- **Early Register-Transfer Level (RTL) design verification methods for timing side-channel attacks.** While vulnerability to timing side-channel attacks is mostly addressed after manufacturing, this work proposes presents an early design phase analysis using simulation-based and formal verification methods. The latter combines the information flow tracking with a commercial formal verification tool, Jasper Gold SPV, and uses a crypto core RSA as a case study. The new method successfully identifies all the timing channels inside the design and provides a lightweight mitigation technique to eliminate the vulnerability. A simulation-based method is developed and applied to an Error Correction Code based decoder as

a part of the fuzzy extractor for memory-based Physical Unclonable Function enabled integrated circuits. The method involves a structural analysis to identify the implementation parameters that might create vulnerabilities to a defined novel attack. The method is validated by simulation.

- **The study of dependencies between side-channel security and soft-error reliability in a design implementation.** The design verification for multiple extra-functional aspects is challenging due to their mutual influence. This work targets the security and reliability aspects of Binary Neural Network (BNN) inference engines used in critical systems and demonstrates the conflicts between security and reliability requirements. The research work studies a recently introduced power side-channel mitigation technique to the BNN hardware implementation and estimates its impact on soft-error reliability of the involved design components. The research has proposed a soft-error reliability analysis flow for BNN hardware inference engines based on logical and functional de-rating. The final results have shown a significant increase in vulnerability to soft errors caused by the security mitigation techniques. In particular, for the case study, there is an up to 1000$\times$ output bit failure rate increase and 350$\times$ neural network functional failure rate increase.

Kokkuvõte
Riistvaraprojektide turva- ja töökindlusaspektide ekstrafunktsionaalse verifitseerimise lähenemisviisid

Elektroonikas võib märgata mitmeid uusi trende. Esiteks võimaldab tehnoloogia miniaturiseerumine keerukate funktsioonide integreerimist üliväikesel pinnal. Sellised kõrge tihedusega kiibid mahutavad miljardeid transistore, muutes need tundlikuks keskkonnamõjude ja rikete suhtes. Nimetatud asjaolu põhjustab väljakutseid funktsionaalsete ja ekstrafunktsionaalsete aspektide verifitseerimisel. Teiseks on koos asjade interneti ja kriitiliste süsteemide populaarsuse tõusuga kaasaegsetes elektroonikasüsteemides hakatud üha enam tähelepanu pöörama ekstrafunktsionaalsetele aspektidele nagu töökindlus, turvalisus ja võimsus ja need on saanud olulisteks parameetriteks projekteerimise funktsionaalse korrektsuse tagamiseks ohutuskriitilistes keskkondades. Kolmandaks on väga suurte ja keerukate kiibiprojektidega toimetulemiseks laialdaselt kasutusele võetud tuumade taaskasutuse metoodika, et suurendada projekteerimisprotsessi tõhusust. Seega on taaskasutatavate tuumade puhul esile kerkinud intellektuaalomandi kaitse ja riistvara troojalastega seotud turvaprobleemid ning need on äratanud ka mikroelektroonika kogukonna tähelepanu.

Verifitseerimistehnikaid tuleb jätkuvalt arendada ning viia kooskõlla ekstrafunktsionaalsete aspektide kõrgete nõudmistega. Kahjuks on verifitseerimistehnikad, mis aitavad käsitleda riistvaraprojektide ekstrafunktsionaalseid aspekte, veel mittepiisavad ning jäävad maha tarkvaradistsipliini vastavast arengust. Tavaliselt koostatakse ekstrafunktsionaalsed aspektid funktsionaalse käitumisjada põhjal. Mõnede ekstrafunktsionaalsete aspektide, nagu töökindlus ja turvalisus, jaoks vajavad äärmiselt abstraktsed ekstrafunktsionaalsed nõuded tohutut analüüsi, et eraldada funktsionaalne järjestus ning see eeldab skeemi põhjalikku ja detailset tundmist. Mõningaid ekstrafunktsionaalseid funktsioone, nagu turvaaspektiga seotud külgkanalite rünnakud, verifitseeritakse tavaliselt alles peale tootmist, mis põhjustab suuri projekteerimiskulusid. Mitme ekstrafunktsionaalse aspekti samaaegsel verifitseerimisel muutub probleem veelgi keerukamaks ning see keerukus suureneb eksponentsiaalselt.Seega on lõputöö eesmärkideks uurida põhjalikult ekstrafunktsionaalseid aspekte ning käsitleda olemasoleva riistvara verifitseerimise metoodika teatud piiranguid ja omadusi, uurides põhjalikult turvalisuse ja töökindluse olulisi ekstrafunktsionaalseid aspekte. Vastavalt sellele võib käesoleva uurimistöö tulemused kokku võtta järgmiselt.

- **Igakülgne arusaam asjakohaste verifitseerimismetoodikate ekstrafunktsionaalsetest aspektidest ja omadustest.** Uurimistöö põhineb ekstrafunktsionaalsete aspektide verifitseerimise teadustöö taseme kirjanduse ülevaatel ja pakub välja ekstrafunktsionaalsete aspektide taksonoomia. Uuring toob välja väljakutsed praeguse riistvara verifitseerimisel ja pakub välja esialgse meetodi, mis hõlmab masinõppe tehnikaid, et tulla toime ekstrafunktsionaalsete funktsioonide analüüsiga.

- **Varajased registersiirde taseme verifitseerimise meetodid ajastuse külgkanali rünnakutele.** Kuigi haavatavust ajastuse külgkanalite rünnakute suhtes käsitletakse enamasti pärast tootmist, pakutakse selles töös välja varase projekteerimisetapi analüüs, kasutades simulatsioonipõhiseid ja formaalseid verifitseerimismeetodeid. Viimane ühendab infovoo monitooringu kommertsiaalse verifitseerimistööriistaga Jasper Gold SPV ja kasutab juhtumiuuringuna krüptotuuma RSA-d. Uus meetod tuvastab edukalt kõik disaini sees olevad ajastuskanalid ja

pakub haavatavuse kõrvaldamiseks lihtsa leevendustehnika. Arendati välja simulatsioonil põhinev meetod ja rakendati seda veaparanduskoodil põhineval dekoodril mälupõhiste füüsikalise unklooneerimata funktsiooniga integraallülituste häguse ekstraktori osana. Meetod hõlmab struktuurianalüüsi, et tuvastada rakendusparameetrid, mis võivad tekitada haavatavusi määratletud uudse rünnaku suhtes. Meetod on valideeritud simulatsiooni abil.

- **Kõrvalkanali turbe ja pehmete vigade töökindluse vaheliste sõltuvuste uurimine projekteerimislahenduses**. Mitme ekstrafunktsionaalse aspekti disaini koos verifitseerimine on nende vastastikuse mõju tõttu keeruline. Käesolev töö on suunatud kriitilistes süsteemides kasutatavate Binaarsete Närvivõrkude (BNN) turvalisuse ja töökindluse aspektidele ning demonstreerib vastuolusid turbe- ja töökindlusnõuete vahel. Uurimistöös uuritakse hiljuti BNN-i riistvararakenduses kasutusele võetud võimsuse külgkanali leevendamise tehnikat ja hinnatakse selle mõju kaasatud disainikomponentide pehmete vigade töökindlusele. Uurimistöös on välja pakutud pehmete vigade usaldusväärsuse analüüsi voog BNN-i riistvara realisatsioonidele, mis põhineb loogilisel ja funktsionaalsel tegurite vähendamisel. Lõpptulemused näitavad, et turvalisuse leevendamise tehnikatest põhjustatud pehmete vigade haavatavus on oluliselt suurenenud. Eelkõige täheldati juhtumiuuringu puhul kuni 1000-kordne väljundbiti tõrkesageduse suurenemine ja 350-kordne närvivõrgu funktsionaalsete rikete sageduse kasv.

Appendix 1

I

M. Jenihhin, X. Lai, T. Ghasempouri, and J. Raik, “Towards multidimensional verification: Where functional meets non-functional,” in *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–7, IEEE, 2018

Towards Multidimensional Verification: Where Functional Meets Non-Functional

Maksim Jenihhin, Xinhui Lai, Tara Ghasempouri, Jaan Raik
Computer Systems, Tallinn University of Technology, Estonia, maksim@ati.ttu.ee

***Abstract*— Trends in advanced electronic systems' design have a notable impact on design verification technologies. The recent paradigms of Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) assume devices immersed in physical environments, significantly constrained in resources and expected to provide levels of security, privacy, reliability, performance and low power features. In recent years, numerous extra-functional aspects of electronic systems were brought to the front and imply verification of hardware design models in multidimensional space along with the functional concerns of the target system. However, different from the software domain such a holistic approach remains underdeveloped. The contributions of this paper are a taxonomy for multidimensional hardware verification aspects, a state-of-the-art survey of related research works and trends towards the multidimensional verification concept. The concept is motivated by an example for the functional and power verification dimensions.**

***Keywords*— extra-functional verification; functional verification; survey; taxonomy; security verification; reliability verification; power verification; timing verification.**

I. Introduction

Today, several prominent trends in electronic systems design can be observed. The Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) devices are immersed in physical environments, significantly constrained in resources and expected to provide levels of security and privacy [1], ultra-low power feature or high performance. Very complex electronic systems, including those built from the non-certified for reliability Commercial-Off-The-Shelf (COTS) components, are used for safety- and business-critical applications. These trends along with gigascale integration at nanoscale technology nodes and multi-/many-processor based systems-on-chip architectures have ultimately brought to the front various *extra-functional aspects* of the electronic systems' design at the chip design level. The latter include security, reliability, timing, power consumption etc. There exist numerous threats causing an electronic system to violate its specification. In the hardware part, these are design errors (bugs), manufacturing defects and variations, reliability issues such as soft errors and aging faults or malicious faults, such as security attacks. Eventually, there can also be bugs in the software part.

Hardware design model verification detects *design errors* affecting *functional* and *extra-functional* (also interchangeably referred as *non-functional*) aspects of the target electronic system. Strictly, the sole *task of extra-functional verification* of a design model is limited to detecting deviations that cause violation of extra-functional requirements. In practice, it often intersects with the task of functional verification [2], [14], thus establishing *a multidimensional space for verification.* A "grey area" in distinction between functional and extra-functional requirements may appear when an extra-functional requirement is a part of design's main functionality. E.g., security requirements for some HW design can be split into extra-functional and functional sets if the design's purpose and specified functionality is a system's security aspect, e.g. it is a secure cryptoprocessor.

In this paper, we present an overview for the recent trends in extra-functional and functional verification of HW designs and discuss the challenges towards the holistic multidimensional verification. The rest of this paper is organized as follows: Section II provides a taxonomy of multidimensional verification aspects, Sections III proposes a state-of-the-art survey with the key trends in verification for the main extra-functional aspects, Section IV discusses the multidimensional verification paradigm and presents a motivational example for the functional and power verification dimensions, Section V draws the conclusions.

II. Taxonomy of Multidimensional Verification Aspects

In practice, relevance of each functional and extra-functional aspect strongly depends on design type, target system application and specific user requirements. Following the design paradigm shift, a number of extra-functional aspects have recently received significant academic research attention e.g., security. At the same time, there already exist established industrial practices for measuring and maintaining separate design qualities, e.g. the RAS (Reliability-Availability-Serviceability) aspect introduced by IBM [6]. While in the software engineering discipline, the taxonomy of extra-functional requirements has a comprehensive coverage by the literature [7]-[12], it cannot be directly re-used for the HW verification discipline because of significant difference in the design models.

Fig 1 introduces a taxonomy of multidimensional verification aspects derived from the performed literature review. The conventional *functional concerns* are *safety and liveness properties*, *combinational and temporal dependencies* along with *data types*, however this list can be extended for particular designs. The extra-functional aspects can be strictly categorized into three groups: *system qualities, system resource constraints* and *timing aspects* (in bold). Despite the *security* and *reliability* aspects belong to the first group and the *power* aspect belongs to the second group, these three aspects have a special attention in the literature and in practical applications. Several extra-functional aspects such as *(manufacturing defects) testability*, *fault-tolerance* and other in-field fault group aspects do not have a direct correspondence in the software engineering discipline because of the distinct nature of faults. Other aspects such as *real-time constraints* are very similar between the two domains.

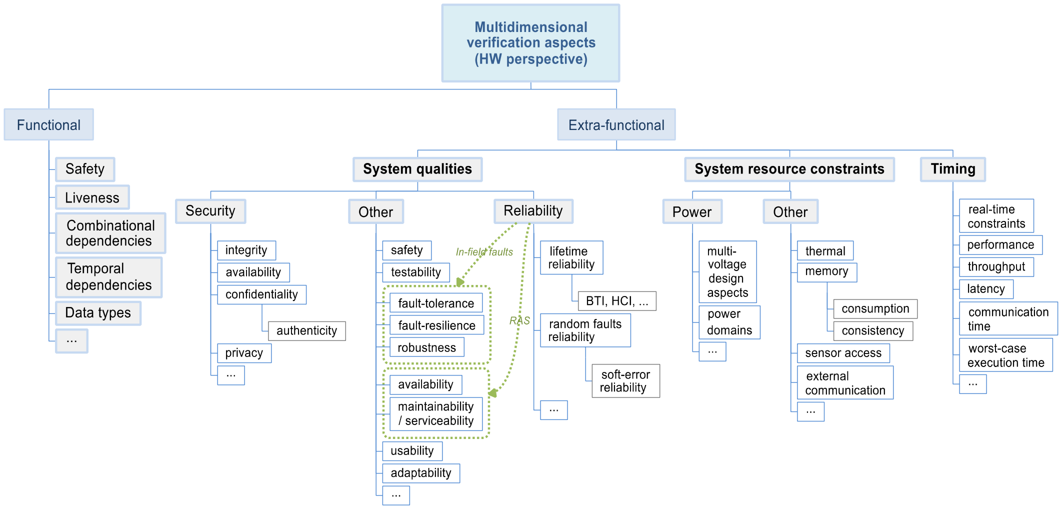


Fig. 1. Taxonomy of multidimensional verification aspects

III. Trends in Extra-Functional Verification

Table I presents a survey of recent publications targeting extra-functional and multidimensional verification. Here, along with the specific extra-functional aspects details about the design model and verification approach are outlined, i.e., the design under verification type, verification engine, the level of abstraction, design representation language, compute model and the tool operated in the research. For instance, the row for paper [40] shows that the authors performed model checking to reduce the state space of a Timed Petri Net of a real-time scheduler. Looking at this row, real-time constraints is the type of timing property, a scheduler of an embedded system is the design under verification, the abstraction level is the system level and SMT model checker Promela [64], Timed Petri Net and SPIN [63] are the verification engine, the design representation language, the compute model and the tool, respectively. We pointed out such key points for all the recent up to 10-year old studies in this area.

A. *Security aspects*

Security is difficult to quantify as today there are no commonly agreed metrics for this purpose [1]. The key targeted security services [16] are commonly represented as non-functional aspects for verification are *confidentiality*, *integrity* and *availability*. They are tightly linked to the type of attack and the attacker model assumed for each case, i.e. black-, grey- or white-box.

Today, for complex HW designs (e.g. IEEE1687 Reconfigurable Scan Networks or NoCs) the specific on-chip security features in the design model to be verified also tend to be very sophisticated. These include on-chip mechanisms for attack prevention (firewalls, user management, communications isolation), attack protection (traffic scrambling, encryption) and attack resilience (checkers for side-channel attacks, covert channel detection, attack recovery mechanisms).

Many of the existing works in security verification (e.g. [21], [23], [25], [28], [29]) are focusing on the integrity attribute, mostly addressing HW trojan detection . There also exist some works that additionally target ([19], [20], [22], [24], [30]) or are exclusively considering ([26], [27]) the confidentiality aspect.

Several solutions in security verification are restricted to target specific architectures or types of modules such as Reconfigurable Scan Networks (RSNs) [22], [26] or macro-asynchronous micro-synchronous pipelines [29].

There is virtually no work that considers security in combination with other extra-functional aspects. Some solutions in the security verification of NoCs indirectly address reliability due to the fact that they implement hardware monitors that allow avoiding both, attacks and in-field faults [20], [21]. An approach that is designed for modeling a multitude of extra-functional aspects is the model-based engineering example of Architecture Analysis and Design Language (AADL) [19]. While, in principle, AADL allows representing several extra-functional aspects (called quality attributes in AADL), [19] only concentrates on analysis of confidentiality as a part of verifying security in a system with multiple levels of security. The authors in [70] have target a general multi-view HW modeling and verification approach taking into consideration the security view.

B. *Reliability aspects*

The key drivers for the reliability aspect in today's designs are the recent industrial standards in different application domains such as IEC61508, ISO26262, IEC61511, IEC62279, IEC62061, RTCA/DO-254, IEC60601, etc. These ultimately imply extra-functional features such as safety mechanisms and redundancy to ensure levels of fault coverage, e.g. ASIL (Automotive Safety Integrity Level). Here, the key threats are transient faults in the field such as radiation-induced single event effects or *soft errors* [15] and intermittent to permanent faults by process or time-dependent variations, i.e. *aging*, e.g. induced by Bias Temperature Instability (BTI) [13]. New applications, demand the systems to be fail-safe or fail-operational, by functionally redundant design parts enabling fault-tolerance, -resilience and -robustness. A promising initiative in reliability specification and modelling is the Reliability Information Interchange Format (RIIF) [30].

TABLE I. SURVEY OF THE STATE-OF-THE-ART SOLUTIONS FOR EXTRA-FUNCTIONAL AND MULTIDIMENSIONAL VERIFICATION

Paper	Year[1]	Extra-functional aspect[2]						Design under verification	Verification engine	Abstract. level[5]	Design representation language	Compute model	Tool
		Security	*Reliability*[3]	*Timing*[4]	*Power*	*Other system quality*	*Other constrained resource*						
[19]	2009	confidentiality, integrity	-	-	-	-	-	HW/SW system	formal, correct-by-construction	SL	AADL	-	OSATE
[20]	2016	integrity, confidentiality	○	-	-	-	-	NoC	simulation, HW monitors	RTL	VHDL/Verilog	-	-
[21]	2014	integrity	○	-	-	-	-	NoC	formal	GL	VHDL/Verilog	-	SurfNoC
[22]	2017	integrity, confidentiality	-	-	-	-	-	RSN	model check	RTL	ICL	Craig interpolation	CIP solver
[23]	2015	integrity	-	-	-	-	-	SoC	simulation	RTL	VHDL/Verilog	-	-
[24]	2016	integrity, confidentiality	-	-	-	-	-	ALU	equivalence check	GL	-	QBF-SAT	-
[25]	2017	integrity	-	-	-	-	-	SoC	semiformal	GL	-	-	JasperGold SPV
[26]	2016	confidentiality	-	-	-	-	-	RSN	model check	RTL	ICL	Craig interpolation	CIP Solver
[27]	2017	confidentiality	-	-	-	-	-	industrial control systems	formal	SL	ASLan++	-	CL-AtSe
[28]	2017	integrity	-	-	-	-	-	IP cores	semiformal	GL	VHDL	-	mini-SAT
[29]	2015	integrity	-	-	-	-	-	ISA, pipeline	model check	RTL	-	CTL, LTL	nuXmv SMV
[30],[71]	2013	integrity, confidentiality	-	-	-	-	-	IPs and SoCs	formal	RTL, GL	Verilog	-	JasperGold SPV
[33]	2017	-	●	-	-	-	-	CPS	model check	SL	AADL	Timed Automata	UPPAAL
[34]	2015	-	SER	-	-	-	-	IP cores	formal	GL/RTL	LDDL	LDDL	Coq
[35]	2010	-	SER	-	-	availability, serviceability	-	processor	fault inject.	GL	Verilog	-	IBM in-house
[36]	2016	-	●	-	-	availability, serviceability	-	SoC	fault inject.	RTL	-	-	-
[38]	2018	-	-	latency	○	-	-	NoC	fault inject.	RTL	VHDL	-	QoSinNoC
[39]	2011	-	-	RT	-	-	-	memory	model check	RTL	REAL;AADL	-	Ocarina
[40]	2010	-	-	RT	-	-	-	Scheduler of a RT emb. system	model check	-	Promela	Time Petri-net	SPIN
[41]	2010	-	-	latency	-	-	-	RT emb. system	model check	SL	AADL	-	YICES
[42]	2017	-	-	performance	○	-	-	NoC, HW/SW architectures	simulation	SL	GAL (Graph Assembly Language)	resource / connectivity graphs	ArchOn
[43],[44]	2016	-	○ (LTR)	-	○	-	thermal	Smart Systems	simulation	SL	IP-XACT; SystemC-AMS	-	-
[46]	2012				●	-	-	IPs	simulation	SL	SystemC	-	-
[47]	2016				●	-	-	DSP cores	simulation	SL,GL,RTL	SystemC	-	Powersim
[50]	2017	-	-	performance	○	-	-	automotive CPS	model check	SL	C, EAST-ADL	Timed Automata	UPPAALsdv
[51]	2016	-	-	-	●	-	-	IPs	semiformal ABV	RTL	VHDL/Verilog; SystemC	Hidden Markov Model	-
[52]	2012	-	-	execution time	○	-	-	distributed emb. system	simulation	SL	SystemC	-	-
[53]	2016	-	-	performance	○	-	thermal	HW/SW platform	simulation, formal (analytical)	RTL,TLM,SL	UML; C++; SystemC-AMS; VHDL	HIF	HIFSuite
[54]	2014	-	-	-	-	connectivity	-	SoC	symbolic model checking	RTL/TLM	Verilog	-	Incisive Formal Verifier
[55]	2008	-	-	○ (latencies)	-	connectivity	-	SoC	property checking	RTL	Verilog	-	JasperGold CV
[56]	2016	-	-	-	-	-	memory consistency	processor	simulation	ISA	ruby	-	McVerSi
[60]	2011	-	-	-	●	-	thermal	SoC	simulation	SL, GL/RTL	SystemC	-	PowerMixer, PowerDepot, PowerBrick,
[61]	2015	-	-	-	●	-	-		simulation	SL,TLM	SytemC	-	Power Kernel Tool
[62]	2011	-	-	-	●	-	-	SoC	simulation	SL	SystemC	-	Powersim
[67]	2018	-	●	-	-	-	-	CPS	formal and simulation, HW monitors	RTL	VHDL/Verilog	multiple	multiple
[68]	2014	-	SER	-	-	-	-	IPs	SAT solver	RTL	VHDL	-	-
[69]	2010	-	SER	-	-	-	-	IPs, processor	simulation	RTL	VHDL/Verilog	-	-
[70]	2018	●	●	-	-	-	-	MPSoC	model check	SL, RTL	-	Timed Automata	UPPAAL

[1] *only conference, journal and industrial white papers published in the last 10 years were selected for this survey*

[2] ● *– this aspect is the main focus in the paper;* ○ *– this aspect is partially addressed*

[3] *LTR – lifetime reliability; SER – soft-error reliability;*

[4] *RT – real-time constraints;*

[5] *GL – gate level; SL – system level; ISA – instruction set architecture level; TLM – transaction level model*

Similar to other aspects, reliability in large complex electronic systems, e.g. safety-critical CPSs is tackled starting at high level of abstraction. System's fault tolerance is formally checked using UPPAAL and timed automata models generated from AADL specifications [33]. HW design models and tools at such a level also enable verification of interference of extra-functional design aspects [70].

There are research works relying on design soft-error reliability verification by fault-injection campaigns e.g. [69] or formal analysis [68]. This analysis is targeted at extra-functional structures for error protection, e.g. error-correction code (ECC) based mechanisms against single-bit errors in memory elements [68]. [34] proposes a general approach to verify gate-level design transformations for reliability against single-event transients by soft errors that combines formal reasoning on execution traces. [35] and [36] focus on the RAS (reliability, availability and serviceability) group of extra-functional aspects outlined by IBM for complex processor designs where embedded error protection mechanisms and designs intrinsic immunity (due to various masking) to errors is evaluated by fault injection.

[43] and [44] propose extensions to system descriptions in the IP-EXACT format to enable multi-layer representation and simulation of several mutually influencing extra-functional aspects of smart system designs such as lifetime reliability (aging), power and temperature. A complex approach to verification of multiple reliability concerns (soft errors, BTI, etc.) across layers in industrial CPS designs is proposed in [67] as a collaborative research result in the IMMORAL project.

C. *Timing aspects*

Functional temporal properties are essential part of sequential designs' specification that are often modelled for functional verification by computational tree logic (*CTL*), applied for formal approaches, and linear temporal logic (*LTL*) temporal assertions expressed arbitrarily in *PSL* (Property Specification Language), SVA (System Verilog Assertions) or systematically in UVM (Universal Verification Methodology). In the extra-functional domain, these can be extended to specific requirements about *performance* (in particular as a trade-off to the power aspects), *quality of service* parameters such as *latency*, *throughput* etc. For formal classification, particular timing aspects may stay in the "limbo" between functional and extra-functional dimensions when timing properties are indivisible with the functionality for the real-time systems (e.g. demanding a worst-case execution time) [32] or time-constrained communication implementations as in the Network-on-Chip (NoC) structures [37], [38].

Several works have been widely studying system's timing properties. Some researchers are mainly focused on generating timing properties such as Real Time (RT), latency, execution time, throughput, communication time, performance and etc., to reduce the verification process, state space and cost [40], [42], [53]. Other works instead use the timing properties to assess whether the system under verification is correctly functioning or not [41], [50], [52]. In [42] a framework has been developed to analyze performance of a system design. The framework is based on stochastic modeling and simulation and it is applied on a set of NoC topologies. The methodology uses the selective abstraction concept to reduce complexity. [53] introduces a tool called CONTREX to complement current activities in the area of predictable computing platforms and segregation mechanisms with techniques to compute RT properties. CONTREX enables energy efficient and cost aware design through analysis and optimization of properties such as RT. In [41], an analysis tool is developed to work with the AADL [65] developing environment to analyze the latency of the AADL model to assure the correctness of a scheduling model that binds the relation of different components in a model. The authors in [50] modified EASTADL [66] to include energy constraints and transformed energy-aware real-time (ERT) behaviors modeled in EAST-ADL/Stateflow into UPPAAL models amenable to formal verification. And finally, in [52] a platform has been developed to generate a virtual platform in SystemC to express the accuracy of real-time embedded system.

A few works also take into account dependencies between several extra-functional aspects. For instance, the work in [50], [53] and [42] present the effect of optimizing timing properties (performance and latency) on power consumption or the study in [52] performs the effect of decreasing execution time on power consumption. Such analysis is mostly limited to two extra functional aspects or neglected at all [39], [40], [41], [55], while design timing constraints can strongly influence not only power consumption but reliability, security, availability, etc. as well as functional properties.

D. *Power consumption*

This extra-functional aspect has a tight relation to the implementation technology assumed for the synthesis of the design model under verification. With planar bulk MOSFET technology known for exponential growth of the static leakage power for smaller device geometries and employment of FinFET and Tri-Gate-Transistors in the advanced technology nodes, the CMOS device parameters are essential for this analysis [45].

In commercial flows, this verification dimension can be addressed relatively independently from the functional verification dimension. The *power intent* and detailed power modelling can be done starting at TLM or RTL with minimal interference with the HDL functional description, e.g. using the Accellera introduced Unified Power Format (UPF) employed for power-aware design verification automation by commercial tools especially with the latest UPF3.0 [48] or Cadence/Si2 Common Power Format CPF [49]. For the advanced device implementation technologies, power specification implies *multi-voltage design* with up to tens of *power domains* and may consider dynamic and adaptive voltage scaling.

In the recent research works, design verification against the power aspect is performed at different abstraction levels with a trade-off between speed and accuracy. Some works such as [46], [47], [61], [62] perform power analysis at system level targeting high simulation speed and low power optimization flexibility similar to the accuracy achievable at lower levels. In [46], the authors applied their approach to SRAM and AES encryption IPs and obtained a significant simulation speed-up in comparison to gate-level simulation with a high fidelity of the system-level power simulation. A promising software tool for power simulation in SystemC designs is the Powersim framework [47], [62]. In [47], a methodology to estimate the dissipation of energy in hardware at any level of abstraction is proposed. In [62], the authors propose a SystemC class library aimed at calculation of energy consumption of hardware described at system level. The work in [60] introduces a series of tools which can be tightly linked and enable the power analysis from layout, gate-level, RT-level, IP-level to system level. The power aspect verification could benefit from a holistic multi-level modelling, such as e.g. [17] available for functional verification. [42], [43], [44], [50], [52], [53], are aimed at methodologies suitable for specific applications (such as cyber-physical system [50]) that assume verification of extra-functional aspects such as power, timing, thermal at the system level.

IV. The Challenge of Multidimensional Verification

The performed analysis of the state of the art has outlined a gap in methodologies and tools for holistic multidimensional verification of hardware design models.

Different from functional verification, approaches for extra-functional hardware design aspects' verification remain underdeveloped even when tackled in isolation. Here, one of the key issues is a lack of established metrics for verification confidence. For a particular functional verification plan, the functional dimension usually includes conventional structural (code) coverage metrics, functional coverage [3] in form of asserted and assumed properties and design parameters along with stimuli quality assessment by model mutations [18]. The metrics for confidence in extra-functional dimension verification results may be challenging as in practice the requirements are *subjective* and can be specified as a mixture of *quantitative* and *qualitative constraints*. Accurate hardware verification in a particular dimension requires both sufficient extra-functional design modeling and the extra-functional aspect target modeling [70]. There is a limited number of dedicated commercial tools and common standards for extra-functional verification flows. In particular, for the security dimension the JasperGold SPV [71] is one of the few such tools that stand out from the academic research frameworks. Finally, the issue of eliciting the extra-functional requirements [4], [5] is a challenging task as ambiguity and (sometimes conflicting) interdependency of the extra-functional aspects in the specifications increases complexity and may leave gaps in the multidimensional verification plans.

Unfortunately, there is no established hardware design methodology supporting multidimensional verification plans for mutually influencing functional and extra-functional aspects. There is a very limited number of research works going beyond analysis of one extra-functional verification aspect under constraints of another as the complexity of the problem grows extremely fast with the number of dimensions (interdependent constraints) and the electronic system size. The first works in this direction are, for example, [44] and [70].

The objective for the research community is to manage multidimensional verification campaigns as illustrated in Fig.2. Fig. 2a is an illustration of six independent verification campaigns in a three-dimensional verification space. Here, a verification campaign can achieve a level of confidence in one, two or all dimensions - (F)unctionality, (P)ower and (S)ecurity. Radar-charts are an instrument for summarizing multidimensional verification results for unlimited number of dimensions, see Fig. 2b (where the dimensions can be ordered to emphasize correlation or interdependencies between adjacent dimensions).

A. Motivational Example

Single-dimension verification campaigns ignoring interdependencies between the dimensions may lead to gaps in the overall electronic system quality. As an example, let us consider an actual verification campaign of an open-source NoC framework Bonfire [57], [58].

The design under verification is in RTL VHDL and implements a 2x2 NoC infrastructure (processing elements excluded). The verification plan considered 2-dimensional verification campaign targeting *functionality* and *power consumption* requirements. For the former, assertion-based

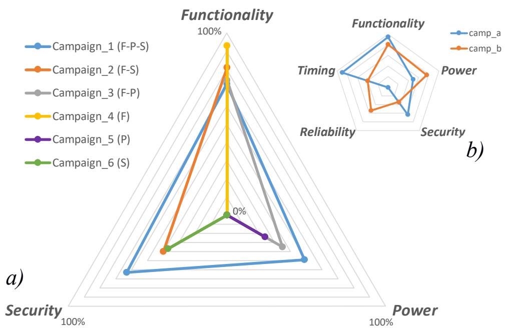


Fig. 2. Multidimensional verification campaigns (Radar-chart *n*-dimensional visualization)

functional verification by simulation was employed targeting statement, branch, condition and toggle coverage metrics and satisfaction of a set of temporal simple-subset PSL assertions. For the latter, a set of power targets were extracted for the targeted silicon implementation assuming a predetermined switching activity.

Among documented design errors, the bug *f1,* as shown in Fig. 3*,* is an example of a functional misbehavior due to improper usage of write and read pointers in the FIFO. The bug *p1* as shown in Fig. 4*,* causes violations of specified power consumption targets because of unnecessary excessive use of a fault-tolerance structure related counter. Interestingly, functionality of both the router core and the complete system is not interfered in case of *p1*.

Table II summarizes power consumption for the three cases. Here, the Total Power is composed of the dynamic power, i.e. the Switching Power in the interconnects and the Internal Power in the logic cells, and the insignificant (for the target technology) static leakage power Leak Power. The case *p1* results in double power consumption compared to the correct implementation and violates the power targets in the specification, whereas the power consumption for the *f1 case* remains within the specification. Design verification in a single dimension may lead to a faulty design.

```
process(write_en, write_pointer) begin --write pointer bug
  if write_en = '1' then
  write_pointer_in <= write_pointer(0)&write_pointer(3 downto 1);
  else
    write_pointer_in <= write_pointer;
  end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer bug
  if (read_en = '1' and empty = '0') then
    read_pointer_in <= read_pointer(0)&read_pointer(3 downto 1);
  else
    read_pointer_in <= read_pointer;
  end if;
end process;
```

```
process(write_en, write_pointer )begin --write pointer
  if write_en = '1' then
  write_pointer_in <= write_pointer(2 downto 0)&write_pointer(3);
  else
    write_pointer_in <= write_pointer;
end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer
  if (read_en = '1' and empty = '0') then
    read_pointer_in <= read_pointer(2 downto 0)&read_pointer(3);
  else
    read_pointer_in <= read_pointer;
  end if;
end process;
```

Fig. 3. Bug *f1* and its correction.

```
process(Healthy_packet, reset_counters, healthy_counter_out)
begin
  if reset_counters  = '1' then
     healthy_counter_in <=  (others => '0');
  elsif Healthy_packet = '1' then                      -- Bug p1!
     healthy_counter_in <= healthy_counter_out + 1;
  else
     healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

```
process(Healthy_packet, reset_counters, healthy_counter_out,
        faulty_counter_out) begin
  if reset_counters  = '1' then
     healthy_counter_in <=  (others => '0');
  elsif Healthy_packet = '1' and faulty_counter_out /=
std_logic_vector(to_unsigned(0, faulty_counter_out'length)) then
     healthy_counter_in <= healthy_counter_out + 1;
  else
     healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

Fig. 4. Bug *p1* and its correction.

TABLE II. POWER CONSUMPTION OF THE CORRECTED BONFIRE SYSTEM IMPLEMENTATION AND THE ONE IN PRESENCE OF BUGS *F1* AND *P1*.

Bonfire system Implementation	Switching Power (mW)	Internal Power (mW)	Leak Power (pW)	Total Power (mW)
with *f1* bug	0.783	9.427	7.50e+05	10.211
with *p1* bug	0.757	21.379	6.93e+05	22.137
corrected	0.666	9.518	7.43e+05	10.184

V. CONCLUSION

In the recent years, numerous extra-functional aspects of electronic systems were brought to the front and imply verification of hardware design models in multidimensional space along with the functional concerns of the target system. In this paper, we have presented a taxonomy for multidimensional hardware verification aspects, a state-of-the-art survey of related research works and trends towards the multidimensional verification concept. The performed analysis of the state of the art has outlined a gap in methodologies and tools for holistic multidimensional verification of hardware design models. The concept was also motivated by a case study for the functional and power verification dimensions.

ACKNOWLEDGMENTS

We would like to acknowledge Apneet Kaur and Behrad Niazmand for their help with the case study analysis. This research was supported in part by projects H2020 MSCA ITN RESCUE funded from the EU H2020 programme under the MSC grant agreement No.722325, H2020 TWINN TUTORIAL, by the Estonian Ministry of Education and Research institutional research grant no. IUT19-1 and by European Union through the European Structural and Regional Development Funds.

REFERENCES

[1] I. Verbauwhede, "Security Adds an Extra Dimension to IC Design: Future IC Design Must Focus on Security in Addition to Low Power and Energy," in *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, pp. 41-45, Fall 2017.

[2] W. Chen, S. Ray, J. Bhadra, M. Abadir and L. C. Wang, "Challenges and Trends in Modern SoC Design Verification," in IEEE Design & Test, vol. 34, no. 5, pp. 7-22, Oct. 2017.

[3] A. Piziali, Functional verification coverage measurement and analysis, Springer, 2008

[4] S. Ullah, M. Iqbal and A. M. Khan, "A survey on issues in non-functional requirements elicitation," *Int. Conf. on Computer Networks and Information Technology*, Abbottabad, 2011, pp. 333-340

[5] Cysneiros L.M., Yu E. (2004) Non-Functional Requirements Elicitation. In: do Prado Leite J.C.S., Doorn J.H. (eds) Perspectives on Software Requirements. The Springer International Series in Engineering and Computer Science, vol 753. Springer, Boston, MA.

[6] M. L. Fair et al., "Reliability, availability, and serviceability (RAS) of the IBM eServer z990," in *IBM Journal of Research and Development*, vol. 48, no. 3.4, pp. 519-534, May 2004.

[7] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, "Non-Functional Requirements," in Software Engineering. Kluwer Academic, 2000

[8] P. Singh, and A. K. Tripathi, "Exploring Problems and Solutions in estimating Testing Effort for Non Functional Requirement," International Journal of Computers & Technology, vol. 3, no 2b, pp. 284-290, 2012.

[9] E. R. Poort, N. Martens, I. Van de Weerd, and H. Van Vliet, "How architects see non-functional requirements: beware of modifiability," In Requirements Engineering: Foundation for Software Quality, pp. 37-51. Springer Berlin Heidelberg, 2012.

[10] D. Ameller, C. Ayala, J. Cabot, and X. Franch, "How do software architects consider non-functional requirements: An exploratory study," Requirements Engineering Conference (RE), pp. 41-50, 2012.

[11] M. Glinz, "On non-functional requirements," In Requirements Engineering Conference, 2007. RE'07, pp. 21-26, IEEE, 2007.

[12] Motus, L. "Analytical Study of Quantitative Timing Properties of Software" 5th EUROMICRO Workshop on Real-Time Systems, 1993.

[13] M. Jenihhin, G. Squillero, T.S. Copetti, V. Tihhomirov, S. Kostin, M. Gaudesi, F. Vargas, J. Raik, M. Sonza Reorda, L. Bolzani Poehls, R. Ubar, G.C. Medeiros, Identification and Rejuvenation of NBTI-Critical Logic Paths in Nanoscale Circuits. JETTA, 32(3),273–289, June 2016.

[14] J. Bhadra, M. S. Abadir, L. C. Wang and S. Ray, "A Survey of Hybrid Techniques for Functional Verification," in IEEE Design & Test of Computers, vol. 24, no. 2, pp. 112-122, 2007

[15] S. Mukherjee, Architecture Design for Soft Errors, Morgan Kauf. 2008.

[16] A. Ptzmann, M. Hansen, "Anonymity Unlinkability Undetectability Unobservability Pseudonymity and Identity Management", A Consolidated Proposal for Terminology version 0.31, 2008.

[17] R. Ubar et al., "Diagnostic Modeling of Digital Systems with Multi-Level DDs". In: Ubar R., Raik J., Vierhaus H.Th. (eds.) Design and test technology for dependable SoC. 2011, pp. 92-118.

[18] V. Guarnieri et al., "Mutation analysis for SystemC designs at TLM," 2011 12th Latin American Test Workshop (LATW), Porto de Galinhas, 2011, pp. 1-6.

[19] J. Hansson, B. Lewis, J. Hugues, L. Wrage, P. Feiler and J. Morley, "Model-Based Verification of Security and Non-Functional Behavior using AADL," in IEEE Security & Privacy, 2009, pp. 1-1.

[20] T. Boraten, D. DiTomaso and A. K. Kodi, "Secure model checkers for Network-on-Chip (NoC) architectures," 2016 Int. Great Lakes Symposium on VLSI (GLSVLSI), Boston, MA, 2016, pp. 45-50.

[21] H. M. G. Wassel et al., "Networks on Chip with Provable Security Properties," in IEEE Micro, vol. 34, no. 3, pp. 57-68, May-June 2014.

[22] M. A. Kochte, M. Sauer, L. R. Gomez, P. Raiola, B. Becker and H. J. Wunderlich, "Specification and verification of security in reconfigurable scan networks," 2017 22nd IEEE European Test Symposium (ETS), Limassol, 2017, pp. 1-6.

[23] L. W. Kim and J. D. Villasenor, "Dynamic Function Verification for System on Chip Security Against Hardware-Based Attacks," in IEEE Transactions on Reliability, vol. 64, no. 4, pp. 1229-1242, Dec. 2015.

[24] Wei Hu et al., "Imprecise security: Quality and complexity tradeoffs for hardware information flow tracking," IEEE/ACM Int. Conference on Computer-Aided Design (ICCAD), Austin, TX, 2016, pp. 1-8.

[25] A. Nahiyan, M. Sadi, R. Vittal, G. Contreras, D. Forte and M. Tehranipoor, "Hardware trojan detection through information flow security verification," 2017 IEEE International Test Conference (ITC), Fort Worth, TX, 2017, pp. 1-10.

[26] M. A. Kochte, R. Baranowski, M. Sauer, B. Becker and H. J. Wunderlich, "Formal verification of secure reconfigurable scan network infrastructure," 2016 21th IEEE European Test Symposium (ETS), Amsterdam, 2016, pp. 1-6.

[27] M. Rocchetto, N. O. Tippenhauer, "Towards formal security analysis of industrial control systems," ACMA sia Conf. Comput. Commun. Secur., 2017, pp. 114–126.

[28] M. Yoshimura, T. Bouyashiki and T. Hosokawa, "A Hardware Trojan Circuit Detection Method Using Activation Sequence Generations," 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), Christchurch, 2017, pp. 221-222.

[29] F. K. Lodhi, S. R. Hasan, O. Hasan and F. Awwad, "Formal analysis of macro synchronous micro asychronous pipeline for hardware Trojan detection," NORCAS 2015, Oslo, pp. 1-4

[30] Z. Hanna, "Verifying Security Aspects of SoC Designs with Jasper App" (white paper), Jasper Design Automation (Cadence), 2013.

[31] A. Savino, S. Di Carlo, A. Vallero, G. Politano, D. Gizopoulos and A. Evans, "RIIF-2: Toward the next generation reliability information interchange format," *IEEE IOLTS*, 2016, pp. 173-178.

[32] C. Liu and J. Layland. Scheduling Algorithms for Multi programming in a Hard Real Time Environment, J. of ACM, 20, pp. 46-61, 1973.

[33] F. S. Gonçalves, D. Pereira, E. Tovar and L. B. Becker, "Formal Verification of AADL Models Using UPPAAL," 2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC), Curitiba, 2017, pp. 117-124.

[34] D. Burlyaev and P. Fradet, "Formal verification of automatic circuit transformations for fault-tolerance," *2015 Formal Methods in Computer-Aided Design (FMCAD)*, Austin, TX, 2015, pp. 41-48.

[35] B. W. Thompto and B. Hoppe, "Verification for fault tolerance of the IBM system z microprocessor," Design Automation Conference, Anaheim, CA, 2010, pp. 525-530.

[36] S. Kan, M. Lam, T. Porter, J. Dworak, "A Case Study: Pre-Silicon SoC RAS Validation for NoC Server Processor", MTV 2016, pp. 19-24.

[37] S. Avramenko, S. P. Azad, S. Esposito, B. Niazmand, M. Violante, J. Raik, M. Jenihhin, "QoSinNoC: Analysis of QoS-Aware NoC Architectures for Mixed-Criticality Applications," in 21st IEEE Int. Symp. DDECS 2018, pp 1-6.

[38] S. Avramenko et al., "Upgrading QoSinNoC: Efficient Routing for Mixed-Criticality Applications and Power Analysis", in IEEE VLSI-SoC 2018, Verona, pp 1-6.

[39] S. Rubini, F. Singhoff and J. Hugues, "Modeling and Verification of Memory Architectures with AADL and REAL," 2011 16th IEEE International Conference on Engineering of Complex Computer Systems, Las Vegas, NV, 2011, pp. 338-343.

[40] H. Wang, X. Zhou, Y. Dong and L. Tang, "A Hierarchical Verification Procedure of Timed Petri-Net Model for Real-Time Embedded Systems," 2010 2nd International Conference on Information Engineering and Computer Science, Wuhan, 2010, pp. 1-4.

[41] H. Wang, X. Zhou, Y. Dong, L. Tang, "Timing Properties Analysis of Real-Time Embedded Systems with AADL Model Using Model Check", IEEE Int. Conf. on Progress in Informatics and Computing (PIC), pp 1019–1023, 2010.

[42] A. Rafiev, F. Xia, A. Iliasov, A. Romanovsky and A. Yakovlev, "Selective Abstraction for Estimating Extra-Functional Properties in Networks-on-Chips Using ArchOn Framework," 2017 17th International Conference on Application of Concurrency to System Design (ACSD), Zaragoza, 2017, pp. 80-85.

[43] S. Vinco, M. Lora, E. Macii and M. Poncino, "IP-XACT for smart systems design: extensions for the integration of functional and extra-functional models," 2016 Forum on Specification and Design Languages (FDL), Bremen, 2016, pp. 1-8.

[44] S. Vinco, Y. Chen, F. Fummi, E. Macii and M. Poncino, "A Layered Methodology for the Simulation of Extra-Functional Properties in Smart Systems," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 10, pp. 1702-1715, 2017.

[45] P. Khondkar, "Low-Power Design and Power-Aware Verification", Springer 2018.

[46] D. Lorenz et al, "Non-invasive Power Simulation at System-Level with SystemC", PATMOS 2012. LNCS (7606), Springer 2012.

[47] S. Orcioni, et al, "Energy estimation in SystemC with Powersim", Integration, the VLSI Journal, (55), 2016, 118-128.

[48] "ANSI/IEEE 1801-2015 - IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems", March 2016,

[49] Si2 "Common Power Format", v2.1, 2014 [Online]

[50] E. Y. Kang, D. Mu, L. Huang and Q. Lan, "Verification and Validation of a Cyber-Physical System in the Automotive Domain," 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, 2017, pp. 326-333.

[51] A. Danese, G. Pravadelli and I. Zandonà, "Automatic generation of power state machines through dynamic mining of temporal assertions," 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016, pp. 606-611.

[52] J. Zimmermann, S. Stattelmann, A. Viehl, O. Bringmann and W. Rosenstiel, "Model-driven virtual prototyping for real-time simulation of distributed embedded systems," 7th IEEE Int. Symposium on Industrial Embedded Systems (SIES'12), Karlsruhe, 2012, pp. 201-210.

[53] R. Görgen et al., "CONTREX: Design of Embedded Mixed-Criticality CONTRol Systems under Consideration of EXtra-Functional Properties," 2016 Euromicro Conference on Digital System Design (DSD), Limassol, 2016, pp. 286-293.

[54] JasperGold Connectivity Verification App, Cadence, http://www.cadence.com [Online]

[55] S.K. Roy, Top Level SOC Interconnectivity Verification Using Formal Techniques. The 8th Int. Workshop on Microprocessor Test and Verification, Austin, TX, USA, 2008, pp. 63–70.

[56] M. Elver and V. Nagarajan, "McVerSi: A test generation framework for fast memory consistency verification in simulation," 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), Barcelona, 2016, pp. 618-630.

[57] Bonfire project website: https://github.com/Project-Bonfire/ [Online]

[58] S. P. Azad et al., "From online fault detection to fault management in Network-on-Chips: A ground-up approach," 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Dresden, 2017, pp. 48-53.

[59] J. Flich and J. Duato, "Logic-based distributed routing for nocs," IEEE Computer Architecture Letters, vol. 7, no. 1, pp. 13–16, Jan 2008.

[60] S.-C. Fang, C.-C. Weng, C.-K. Tseng, C.-W. Hsu, J.-L. Liao, S.-Y. Huang, C.-L. Lung, D.-M. Kwai, SoC power analysis framework and its application to power-thermal co-simulation, in: 2011 Int. Symp. on VLSI Design, Automation and Test, April 2011, pp. 1–4.

[61] G. Vece, M. Conti, S. Orcioni, Transaction-level power analysis of VLSI digital systems, Integr. VLSI J. 50 (2015) 116–126

[62] M. Giammarini, M. Conti, S. Orcioni, System-level energy estimation with Powersim, in: 2011 18th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS), December 2011, pp. 723–726.

[63] Spin tool website. http://spinroot.com/ [Online].

[64] Smt-comp tool website. http://www.smtcomp.org/ [Online].

[65] The architecture analysis and design language AADL. http://www.aadl.info/aadllcurrentsite/ [Online].

[66] EAST-ADL Consortium, "East-adl domain model specification v2.1.9," Maenad European Project, Tech. Rep., 2011.

[67] G. Aleksandrowicz et al. (2018) Designing Reliable Cyber-Physical Systems. In: Fummi F., Wille R. (eds) Languages, Design Methods, and Tools for Electronic System Design. Lecture Notes in Electrical Engineering, vol 454. Springer, Cham

[68] Eli Arbel, Shlomit Koyfman, Prabhakar Kudva, and Shiri Moran. Automated detection and verification of parity-protected memory elements. In Proc. IEEE/ACM ICCAD, 2014, 1-8.

[69] M. Maniatakos, Y. Makris, "Workload-driven selective hardening of control state elements in modern microprocessors". In VTS, 2010, 159–164

[70] J. Vain, A. Kaur, L. Tsiopoulos, J. Raik and M. Jenihhin, "Multi-view modeling for MPSoC design aspects", IEEE BEC October 8-10, 2018

[71] JasperGold Security Path Verification App, Cadence, http://www.cadence.com [Online]

Appendix 2

II
X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," *Microprocessors and Microsystems*, vol. 71, p. 102867, 2019

Microprocessors and Microsystems 71 (2019) 102867

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Understanding multidimensional verification: Where functional meets non-functional

Xinhui Lai[a,*], Aneesh Balakrishnan[a,b], Thomas Lange[b,c], Maksim Jenihhin[a], Tara Ghasempouri[a], Jaan Raik[a], Dan Alexandrescu[b]

[a] Department of Computer Systems, Tallinn University of Technology, Akadeemia 15A, Tallinn 12618, Estonia
[b] IROC Technologies, 2 Square Roger Genin, 5th floor, Grenoble, 38000, France
[c] Dipartimento di Informatica e Automatica, Politecnico di Torino, Turin, Italy

A R T I C L E I N F O

Article history:
Received 25 February 2019
Revised 28 June 2019
Accepted 5 August 2019
Available online 5 August 2019

Keywords:
Extra-functional verification
Functional verification
Survey
Taxonomy
Security verification
Reliability verification
Power verification
Machine learning

A B S T R A C T

Advancements in electronic systems' design have a notable impact on design verification technologies. The recent paradigms of Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) assume devices immersed in physical environments, significantly constrained in resources and expected to provide levels of security, privacy, reliability, performance and low-power features. In recent years, numerous extra-functional aspects of electronic systems were brought to the front and imply verification of hardware design models in multidimensional space along with the functional concerns of the target system. However, different from the software domain such a holistic approach remains underdeveloped. The contributions of this paper are a taxonomy for multidimensional hardware verification aspects, a state-of-the-art survey of related research works and trends enabling the multidimensional verification concept. Further, an initial approach to perform multidimensional verification based on machine learning techniques is evaluated. The importance and challenge of performing multidimensional verification is illustrated by an example case study.

1. Introduction

Recently, several prominent trends in electronic systems design can be observed. Safety-critical applications in the automotive domain set stringent requirements for electronics certification, the Internet-of-Things (IoT) and Cyber-Physical Systems (CPS) devices are immersed in physical environments, significantly constrained in resources and expected to provide levels of security and privacy [1], ultra-low power feature or high performance. Very complex electronic systems, including those built from the non-certified for reliability commercial-off-the-shelf components, are used for safety- and business-critical applications. These trends along with gigascale integration at nanoscale technology nodes and multi-/many-processor based systems-on-chip architectures have ultimately brought to the front various *extra-functional aspects* of the electronic systems' design at the chip design level. The latter include security, reliability, timing, power consumption, etc. There exist numerous threats causing an electronic system to violate its specification. In the hardware part, these are design errors (bugs), manufacturing defects and variations, reliability issues, such as soft errors and aging faults, or malicious faults, such as security attacks. Withal, there can also be bugs in the software part.

Hardware design model verification detects *design errors* affecting *functional* and *extra-functional* (interchangeably referred as *non-functional*) aspects of the target electronic system. Strictly, the sole *task of extra-functional verification* of a design model is limited to detecting deviations that cause violation of extra-functional requirements. In practice, it often intersects with the task of functional verification [2,14], thus establishing *a multidimensional space for verification*. A "grey area" in distinction between functional and extra-functional requirements may appear when an extra-functional requirement is a part of design's main functionality. E.g., security requirements for some HW design can be split into extra-functional and functional sets if the design's purpose and specified functionality is a system's security aspect, e.g. it is a secure cryptoprocessor.

The contributions of this paper are a taxonomy for multidimensional hardware verification aspects, a state-of-the-art survey of related research works towards enabling the multidimensional verification concept. Further, an approach is evaluated which performs multidimensional verification by using machine learn-

* Corresponding author.
E-mail address: xinhui.lai@taltech.ee (X. Lai).

https://doi.org/10.1016/j.micpro.2019.102867
0141-9331/© 2019 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)

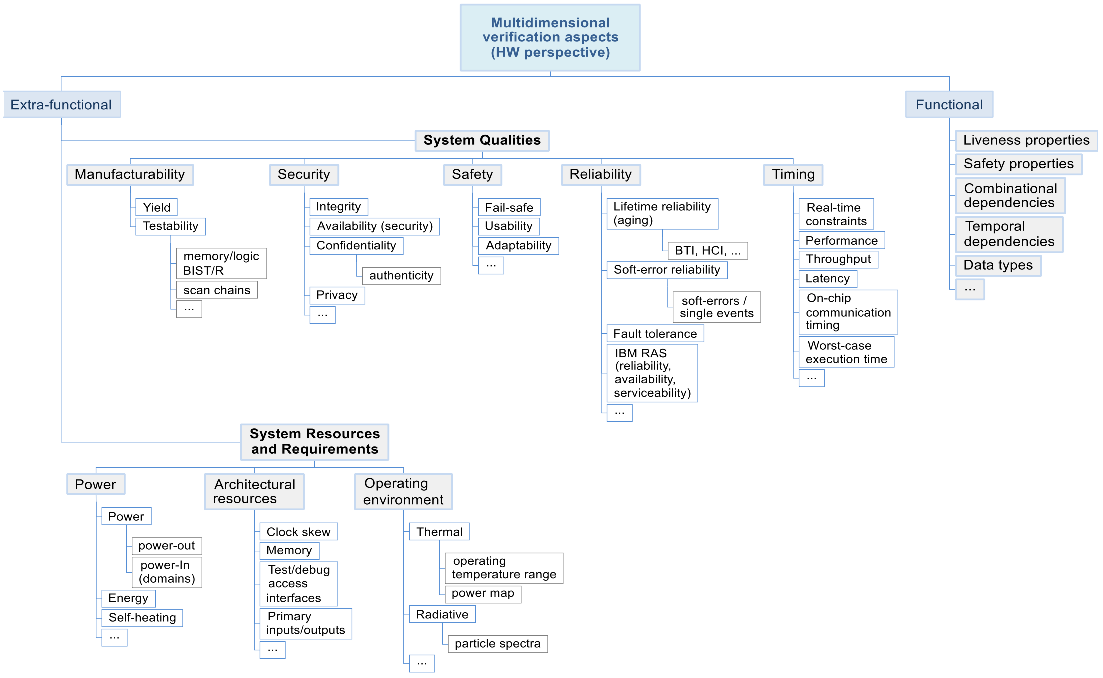


Fig. 1. Taxonomy of multidimensional verification aspects.

ing techniques. The rest of this paper is organized as follows. Section 2 provides a taxonomy of multidimensional verification aspects. Sections 3 proposes a state-of-the-art survey with the key trends in verification for the main extra-functional aspects. Section 4 discusses the multidimensional verification challenges and presents a motivational example for the functional and power verification dimensions. Section 5 proposes adoption of machine learning techniques for support of design's multi-aspect features extraction and verification. Finally, Section 6 draws the conclusions.

2. Taxonomy of multidimensional verification aspects

In practice, relevance of each functional and extra-functional aspect strongly depends on the design type, target system application and specific user requirements. Following the design paradigm shifts, a number of extra-functional aspects have recently received significant academic research attention e.g., security. At the same time, there already exist established industrial practices for measuring and maintaining particular design qualities, e.g. the RAS (Reliability-Availability-Serviceability) aspect introduced by IBM [6]. While in the software engineering discipline, the taxonomy of extra-functional requirements has a comprehensive coverage by the literature [7–12], it cannot be directly re-used for the HW verification discipline because of significant difference in the design models.

Fig 1 introduces a taxonomy of multidimensional verification aspects derived from the performed literature review. The conventional *functional concerns* are *safety and liveness properties, combinational and temporal dependencies* along with *data types*, however this list can be extended for particular designs. The extra-functional aspects can be strictly categorized into two groups: *System Qualities* and *System Resources and Requirements* (in bold). The main system qualities for extra-functional verification are manufacturability of the design, security, in-field safety, reliability during the operational lifespan and a set of timing aspects. The second group embraces the power and architectural resources as well as design constraints set by the operational environment.

Several extra-functional aspects such as *manufacturability*, i.e. primarily *yield* and *testability* against manufacturing defects, *fault-tolerance*, reliability (subject to transient, intermittent and permanent hardware faults) and several aspects from the *System Resources and Requirements* group do not have a direct correspondence in the software engineering discipline because of the distinct nature of faults and specification violations. Other aspects such as *real-time constraints* are very similar between the two domains.

3. Trends in extra-functional verification

Table 1 presents a survey of recent publications targeting extra-functional and multidimensional verification. Here, along with the specific extra-functional aspects details about the design model and verification approach are outlined, i.e., the design under verification type, verification engine, the level of abstraction, design representation language, compute model and the tool operated in the research. We pointed out such key points for all the recent up to 10-year old studies in this area. Further, in the following subsections, we focus on understanding trends for the extra-functional aspects that have the strongest attention in the literature, i.e. security, in-field reliability, timing and power.

3.1. Security aspects

Security is difficult to quantify as today there are no commonly agreed metrics for this purpose [1]. The key targeted security services [16] commonly represented as extra-functional aspects for verification are *confidentiality, integrity* and *availability.* Verifying

Table 1
Survey of the state-of-the-art solutions for extra-functional and multidimensional verification.

Pub.	Year[a]	Extra-functional aspect[b]					Design under verification	Verification engine	Abst. level[e]	Design representation language	Compute model	Tool
		Security	Reliability[c]	Timing[d]	Power	Other aspects						
[19]	2009	confidentiality, integrity	–	–	–	–	HW/SW system	formal, correct-by-construction	SL	AADL	–	OSATE
[20]	2018	confidentiality	–	–	–	–	NoC	unbounded model-checking	RTL	VHDL/Verilog, PSL	LTL	–
[21]	2016	integrity, confidentiality	○	–	–	–	NoC	simulation, HW monitors	RTL	VHDL/Verilog	–	–
[22]	2014	integrity	○	–	–	–	NoC	formal	GL	VHDL/Verilog	–	SurfNoC
[23]	2017	integrity, confidentiality	–	–	–	–	RSN	model check	RTL	ICL	Craig interpolation	CIP solver
[24]	2015	integrity	–	–	–	–	SoC	simulation	RTL	VHDL/Verilog	–	–
[25]	2016	integrity, confidentiality	–	–	–	–	ALU	equivalence check	GL	–	QBF-SAT	–
[26]	2017	integrity	–	–	–	–	SoC	semiformal	GL	–	–	JasperGold SPV
[27]	2016	confidentiality	–	–	–	–	RSN	model check	RTL	ICL	Craig interpolation	CIP Solver
[28]	2017	confidentiality	–	–	–	–	control systems	formal	SL	ASLan++	–	CL-AtSe
[29]	2017	integrity	–	–	–	–	IP cores	semiformal	GL	VHDL	–	mini-SAT
[30]	2015	integrity	–	–	–	–	ISA, pipeline	model check	RTL	–	CTL, LTL	nuXmv SMV
[31]	2018	confidentiality	–	–	–	–	cache	model check	SL	–	CTL	–
[32]	2014	confidentiality	–	–	–	–	cache	model check	RTL/SL	–	FSM	Murphi
[33]	2017	confidentiality	–	–	–	–	cache	model check	RTL/SL	–	FSM	CacheAudit
[34,35]	2013	integrity, confidentiality	–	–	–	–	IPs and SoCs	formal	RTL, GL	Verilog	–	JasperGold SPV
[36]	2018	●	●	–	–	–	MPSoC	model check	SL, RTL	–	Timed Automata	UPPAAL
[41]	2017	–	●	–	–	–	CPS	model check	SL	AADL	Timed Automata	UPPAAL
[42]	2015	–	SER	–	–	–	IP cores	formal	GL/RTL	LDDL	LDDL	Coq
[43]	2010	–	SER	–	–	availability, serviceability	processor	fault inject.	GL	Verilog	–	IBM in-house
[44]	2016	–	●	–	–	availability, serviceability	SoC	fault inject.	RTL	–	–	–
[45,46]	2016	–	○ (LTR)	–	○	thermal	Smart Systems	simulation	SL	IP-XACT, SystemC-AMS	–	–
[47]	2018	–	●	–	–	–	CPS	formal /simulation, HW monitors	RTL	VHDL/Verilog	multiple	multiple

(*continued on next page*)

Table 1 (*continued*)

Pub.	Year[a]	Extra-functional aspect[b]					Design under verification	Verification engine	Abst. level[e]	Design representation language	Compute model	Tool
		Security	Reliability[c]	Timing[d]	Power	Other aspects						
[48]	2014	–	SER	–	–	–	IPs	SAT solver	RTL	VHDL	–	–
[49]	2010	–	SER	–	–	–	IPs, processor	simulation	RTL	VHDL/Verilog	–	–
[50]	2014	–	SER	–	–	–	memory	circuit-level simulation	circuit level	–	–	INFORMER
[51,52]	2018	–	–	comm. constraint	○	–	NoC	fault inject.	RTL	VHDL	–	QoSinNoC
[53]	2011	–	–	RT	–	–	memory	model check	RTL	REAL/AADL	–	Ocarina
[54]	2010	–	–	RT	–	–	Scheduler of RT system	model check	–	Promela	Time Petri-net	SPIN
[55]	2010	–	–	latency	–	–	RT emb. system	model check	SL	AADL	–	YICES
[56]	2017	–	–	performance	○	–	NoC, HW/SW architectures	simulation	SL	Graph Assembly Language	connectivity graphs	ArchOn
[58]	2012				●	–	IPs	simulation	SL	SystemC	–	–
[59]	2016				●	–	DSP cores	simulation	SL,GL, RTL	SystemC	–	Powersim
[62]	2017	–	–	RT	○	–	automotive CPS	model check	SL	C, EAST-ADL	Timed Automata	UPPAALsdv
[63]	2016	–	–	–	●	–	IPs	Semiformal ABV	RTL	VHDL/Verilog; SystemC	Hidden Markov Model	–
[64]	2012	–	–	execution time	○	–	distributed emb. system	simulation	SL	SystemC	–	–
[65]	2016	–	–	performance RT	○	thermal	HW/SW platform	semiformal	RTL,TLM,SL	UML,C++,VHDL SystemC-AMS	HIF	HIFSuite
[66]	2009	–	–	throughput	–	–	SoC/FPGA	simulation	RTL	Verilog/VHDL	–	Modelsim
[67]	2018	–	–	throughput	–	–	NoC	simulation	RTL	System Verilog	–	UVM
[68]	2014	–	–	–	–	connectivity	SoC	symbolic model checking	RTL, TLM	Verilog	–	Incisive Formal Verifier
[70]	2016	–	–	–	–	memory consistency	processor	simulation	ISA	ruby	–	McVerSi
[73]	2011	–	–	–	●	thermal	SoC	simulation	SL,GL, RTL	SystemC	–	Power-Mixer, -Depot, -Brick
[74]	2015	–	–	–	●	–		simulation	SL,TLM	SytemC	–	Power Kernel Tool
[75]	2011	–	–	–	●	–	SoC	simulation	SL	SystemC	–	Powersim

[a] *only conference, journal and industrial white papers published in the last 10 years were selected for this survey.*
[b] *● – this aspect is the main focus in the paper; ○ – this aspect is partially addressed.*
[c] *LTR – lifetime reliability; SER – soft-error reliability;.*
[d] *RT – real-time constraints;.*
[e] *GL – gate level; SL – system level; ISA – instruction set architecture level; TLM – transaction level model.*

security aspects is highly dependent on the type of attack and the attacker model assumed.

Many of the existing works in security verification (e.g. [22,24,26,29,30]) are focusing on the integrity attribute, mostly addressing hardware trojan detection. There also exist works that additionally target [19,21,23,25,34] or are exclusively considering [27,28] the confidentiality aspect. Several solutions in security verification are restricted to specific target architectures or types of modules such as Reconfigurable Scan Networks (RSNs) [23,27] or macro-asynchronous micro-synchronous pipelines [30]. To that end, for complex hardware architectures (e.g. large IEEE1687 Reconfigurable Scan Networks or MPSoCs) the specific on-chip security features to be verified also tend to be very sophisticated. These may include on-chip mechanisms for attack prevention (firewalls, user management, communications' isolation), attack protection (traffic scrambling, encryption) and attack resilience (checkers for side-channel attacks, covert channel detection, attack recovery mechanisms). Several works consider security verification for NoC-based MPSoCs. [20] proposes a method to formally verify the correctness and the security properties of a NoC router. Some solutions in the security verification of NoCs do indirectly address reliability due to the fact that they implement hardware monitors that allow avoiding both, attacks and in-field faults [21,22].

According to recent surveys [37] and [38] cache access driven side-channel attacks have become a major concern in hardware security. In modern processors, deep hierarchy of cache memory is implemented to increase system performance. However, this makes modern computing systems, including IoT devices, vulnerable to cache side-channel attacks. There exist several works addressing verification of the cache security. In [31], the authors propose.

Computation Tree Logic (CTL) based modeling of timing-driven and access-driven cache attacks. This work concentrates on formally describing the attack types. Zhang and Lee [32] models cache as a state machine and proposes a metric based on the non-interference condition to evaluate the access-based cache vulnerability. Canones et al. [33] proposes a model to formally analyze the security of different cache replacement policies. None of the above-mentioned works consider multiple dimensions, or aspects.

An approach that is designed for modeling a multitude of extra-functional aspects is the model-based engineering example of Architecture Analysis and Design Language (AADL) [19]. While, in principle, AADL allows representing several extra-functional aspects (called quality attributes in AADL), Hansson et al. [19] only concentrates on analysis of confidentiality as a part of verifying security in a system with multiple levels of security. The authors in [36] have targeted a general Uppaal Timed Automata based multi-view hardware modeling and verification approach taking into consideration of the security view. The survey of related literature clearly shows that, up to this moment, there is virtually no work considering security verification in combination with other extra-functional aspects.

3.2. Reliability aspects

The key drivers for the reliability aspect in today's designs are the recent industrial standards in different application domains such as IEC61508, ISO26262, IEC61511, IEC62279, IEC62061, RTCA/DO-254, IEC60601, etc. Integrated circuits used in high-reliability applications, e.g. complying with high (Automotive) Safety Integrity Level - (A)SIL, must demonstrate low failure rates (modelled by FIT – Failures in Time) and high fault coverage (e.g. Single-Point Failure Metric SPFM and Latent Fault Metric LFM). These requirements ultimately mandate extra-functional validation efforts for reliability analysis, such as Failure Mode and Effects (Criticality) Analysis - FME(C)A and imply generalized use of methods and features, such as safety mechanisms, for error management. *Functional safety* is a property of the complete system rather than just a component property because it depends on the integrated operation of all sensors, actuators, control devices, and other integrated units. The goal is to reduce the residual risk associated with a functional failure of the target system below a threshold given by the assessment of severity, exposure, and controllability.

The dominant threats for reliability are, first, random hardware faults such as transient faults by radiation-induced single event effects or soft errors [15], i.e. a subject for *Soft-Error Reliability* (SER). Second, these are extreme operating conditions, electronic interference and intermittent to permanent faults by process or time-dependent variations, such as aging induced by Bias Temperature Instability (BTI) [13], where the latter is a subject for *Life-Time Reliability* (LTR). Reliability verification challenge is emphasized by the adoption of advanced nanoscale implementation technology nodes and high complexity of systems, utilizing tens or hundreds of complex microelectronic components and embedding large quantities of standard logic and memory. Moreover, these designs integrate IP cores from multiple design teams making reliability evaluation task to be scattered and complex. Initiatives such as RIIF (Reliability Information Interchange Format) [39], allow the formalization, specification and modeling of extra-functional, reliability properties for technology, circuits and systems.

Similar to other aspects, reliability in large complex electronic systems, e.g. safety-critical CPSs, may be tackled starting at high level of abstraction. System's fault tolerance is formally checked using UPPAAL and timed automata models generated from AADL specifications [41]. HW design models and tools at such a level also enable verification of interference of several extra-functional design aspects [36]. There are research works relying on design soft-error reliability verification by fault-injection campaigns, e.g. [49], or formal analysis, e.g. error-correction code (ECC) based mechanisms against single-bit errors in memory elements [48]. Burlyaev and Fradet [42] proposes a general approach to verify gate-level design transformations for reliability against single-event transients by soft errors that combines formal reasoning on execution traces. Thompto and Hoppe [43] and Kan et al. [44] focus on the RAS (Reliability, Availability and Serviceability) group of extra-functional aspects outlined by IBM for complex processor designs where embedded error protection mechanisms and designs intrinsic immunity (due to various masking) to errors is evaluated by fault injection. Vinco et al. [45,46] propose extensions to system descriptions in the IP-EXACT format to enable multi-layer representation and simulation of several mutually influencing extra-functional aspects of smart system designs such as lifetime reliability, power and temperature. A complex approach to verification of multiple reliability concerns (soft errors, BTI, etc.) across layers in industrial CPS designs is proposed in [47] as a collaborative research result in the IMMORTAL project. Last but not least, addressing the need for reliability verification automation tools, in [50], authors propose a fully automated tool INFORMER to estimate memory reliability metrics by circuit-level simulations of failure mechanisms such as soft-errors and parametric failures.

The survey clearly shows that currently there is a very small number of works considering verification of reliability together with other aspects.

3.3. Timing aspects

Functional temporal properties are essential part of sequential designs' specification that are often modelled for functional verification by Computational Tree Logic (CTL), applied for formal approaches, and Linear Temporal Logic (LTL) temporal assertions expressed arbitrarily, e.g. in Property Specification Language (PSL), System Verilog Assertions (SVA) or systematically, e.g. in Universal

Verification Methodology (UVM). In the extra-functional context, these can be extended to specific requirements and properties such as: *real-time (RT), performance, throughput, latency, on-chip communication time constraint, worst-case execution time* constraints, etc. Several works have been widely studying these timing properties. Some researchers are mainly focused on generating timing properties to reduce the verification effort, for example, state space and cost [54,56,65]. Other works instead use the timing properties to assess whether the system under verification is correctly functioning or not [55,62,64]. In the following, we discuss state of the art for each timing aspect.

A real-time system describes hardware and software systems subject to a *real-time constraint*, that ensures response within a specified time. The correctness of the function depends both on the correctness of the result and also the timeliness of the periods. In [54], an approach to verify the timed Petri-Net model is proposed. A non-instantaneous model is abstracted from the timed Petri-Net model in a hierarchical structure. The non-instantaneous model which is verified with a model-checking tool is used to reduce the state space of the timed Petri-Net model for verification with a satisfiability modulo theories solvers [76,77]. The timed Petri-Net is used to model the interacting relations of the software components and the binding relations between software and hardware in a certain period of time. Görgen et al. [65] introduces a tool called CONTREX to complement current activities in the area of predictable computing platforms and segregation mechanisms with techniques to compute real-time properties. CONTREX enables energy-efficient and cost-aware design through analysis and optimization of real-time constraint. The authors in [62] proposed a method to combine real-rime constraint aspect of a model with energy-aware real-time (ERT) behaviors of the model into UPPAAL for formal verification.

Throughput is a measure of how many units of information a system can process in a given amount of time. In [66], a verification environment has been proposed to estimate the throughput of a SoC. The intention of the paper is to judge whether the verification system can handle SOC verification and provide the necessary performance in terms of speed and throughput. Khamis et al. [67] introduced a Universal Verification Methodology (UVM) environment to measure throughput of a NoC. UVM is a SystemVerilog class library explicitly designed to help and build modular reusable verification components and test-benches. It is an industry standard, so it is possible to acquire UVM IP from other sources and reuse them.

Performance refers to the amount of work which is done during a process, for instance, executing instructions per second. In [56], a framework has been developed to analyze performance of a system design. The framework is based on stochastic modeling and simulation and it is applied on a set of NoC topologies. The methodology uses a selective abstraction concept to reduce complexity.

When referring to hardware, *latency* is the time required for a hardware component to respond to a request made by another component. However, in the cast of hardware, latency is sometimes referred to as the *access time*. In [55], an analysis tool is developed to work with the AADL models [78] to assure the correctness of a scheduling model that binds the relation of different components in a model.

On-chip communication time constraints refer to the requirements on the start and end times of each task in a system critical path, which is the sequence of tasks that cannot be delayed without delaying the entire system. For instance, in [51] and [52] a framework has been proposed, which is based on a set of quality of service aware NoC architectures along with the analysis methodology including selected relevant metrics that enable an efficient trade-off between guarantees and overheads in mixed-criticality application scenarios. These architectures overcome the notion of strictly divided regions by allowing non-critical communication pass through the critical region, providing they do not utilize common router resources. Such problem formulation is relevant to facilitate the usage of NoC technology by safety-critical industries such as avionics.

The *worst-case execution time* of a computational task is the maximum length of time the task could take to execute on a specific hardware platform. The designer of a system can employ techniques such as schedulability analysis to verify that the system responds fast enough [40]. For instance, Zimmermann et al. [64] presents an approach to generate a virtual execution platform in SystemC to advance the development real-time embedded systems including early validation and verification. These virtual execution platforms allow the execution of embedded software with strict consideration of the underlying hardware platform configuration in order to reduce subsequent development costs and to allow a short time-to-market by tailoring and exploring distributed embedded hardware and software architectures.

Last but not least, a few works also take into account dependencies between several extra-functional aspects. For instance, the work in [62,65] and [56] present the effect of optimizing timing properties (performance and latency) on power consumption or the study in [64] performs the effect of decreasing execution time on power consumption. Such analysis is mostly limited to two extra functional aspects or neglected at all [53–55,69], while design timing constraints can strongly influence not only power consumption but reliability, security, availability, etc. as well as functional properties.

3.4. Power aspects

In commercial flows, verification of the power aspect can be addressed relatively independently from the functional verification dimension. The *power intent* and detailed power modelling can be done starting at TLM or RTL with minimal interference with the HDL functional description, e.g. using the Accellera introduced Unified Power Format (UPF) employed for power-aware design verification automation by commercial tools especially with the latest UPF3.0 [60] or Cadence/Si2 Common Power Format CPF [61]. For the advanced device implementation technologies, power specification implies *multi-voltage design* with up to tens of *power domains* and may consider dynamic and adaptive voltage scaling.

In the recent research works, design verification against the power aspect is performed at different abstraction levels with a trade-off between speed and accuracy. Some works such as [58,59,74,75] perform power analysis at system level targeting high simulation speed and low power optimization flexibility similar to the accuracy achievable at lower levels. In [58], the authors applied their approach to SRAM and AES encryption IPs and obtained a significant simulation speed-up in comparison to gate-level simulation with a high fidelity of the system-level power simulation. A promising software tool for power simulation in SystemC designs is the Powersim framework [59,75]. In [59], a methodology to estimate the dissipation of energy in hardware at any level of abstraction is proposed. In [75], the authors propose a SystemC class library aimed at calculation of energy consumption of hardware described at system level. The work in [73] introduces a series of tools (PowerBrick (construct power library for standard cell library), PowerMixer (for RTL/gate-level estimator), PoweMixer^{ip} (IP-based model builder), PowerDepot (estimate system-level power consumption)) which can be tightly linked and enable the power analysis from layout, gate-, RT-, IP- to system level with a good simulation speed while retaining high accuracy. The power aspect verification could benefit from a holistic multi-level modelling, such as e.g. [17] available for functional verification. Rafiev et al.

[56], Vinco et al. [45,46], Kang et al. [62], Zimmermann et al. [64], Görgen et al. [65], are aimed at methodologies suitable for specific applications (such as cyber-physical system [62]) that assume verification of extra-functional aspects such as power, timing, thermal at the system level.

This extra-functional aspect has a tight relation to the implementation technology assumed for the synthesis of the design model under verification. With planar bulk MOSFET technology known for exponential growth of the static leakage power for smaller device geometries and employment of FinFET and Tri-Gate-Transistors in the advanced technology nodes, the CMOS device parameters are essential for this analysis [57].

3.5. Machine learning based techniques

The complex problem of multidimensional verification can be assisted by the recent advances in the machine learning discipline. This type of approaches (along with e.g. evolutionary algorithms) is particularly suitable for multi-aspect optimization problems where formal deterministic approaches may lack scalability.

Machine Learning (ML) is the concept of a machine learning from examples and making predictions based on its experience, without being explicitly programmed [82]. Previous works have shown that ML can be used for verification purposes at different levels. In [83], machine learning was introduced in physical design analysis. The feasibility of ML in physical design verification (e.g., lithography hotspot detection) was investigated, and a reference model for application was presented. Based on this work [84] used ML to increase the speed of the performance evaluation (power and area) of a circuit design after physical design by a factor of 40 as well as performing a Design Rule Check. In [85], ML was used to predict the timing behavior of the final floorplan of a circuit during the Place & Route routine and thus, shifting the analysis to an earlier design stage. In [79], the analysis is moved even to higher abstraction level. The high-level synthesis (HLS) resource usage and timing estimation was improved by train ML models with data from real implementations. Thus, the design flow can be assisted with machine learning and predict accurate values even in very early design stages. Machine learning was further applied for Security Verification in [80,81,86], where it was used to detect Hardware Trojans based on features from the Gate Level Netlist. In Section 5, we propose an approach to assist the multidimensional verification flow by using machine learning techniques to estimate a reliability metric, as well as timing metric.

4. The challenges of multidimensional verification

The performed analysis of the state of the art has outlined a gap in methodologies and tools for holistic multidimensional verification of hardware design models.

Different from functional verification, approaches for extra-functional hardware design aspects' verification remain underdeveloped even when tackled in isolation. Here, one of the key issues is a lack of established metrics for verification confidence. For a particular functional verification plan, the functional dimension usually includes conventional structural (code) coverage metrics, functional coverage [3] in form of asserted and assumed properties and design parameters along with stimuli quality assessment by model mutations [18]. The metrics for confidence in extra-functional dimension verification results may be challenging as in practice the requirements are *subjective* and can be specified as a mixture of *quantitative* and *qualitative constraints*. Accurate hardware verification in a particular dimension requires both sufficient extra-functional design modeling and the extra-functional aspects target modeling [36]. There is a limited number of dedicated commercial tools and common standards for extra-functional verification flows. In particular, for the security dimension the JasperGold SPV [35] is one of the few such commercial tools that stand out from the academic research frameworks. Finally, the issue of eliciting the extra-functional requirements [4,5] is a challenging task as ambiguity and (sometimes conflicting) interdependency of the extra-functional aspects in the specifications increases complexity and may leave gaps in the multidimensional verification plans.

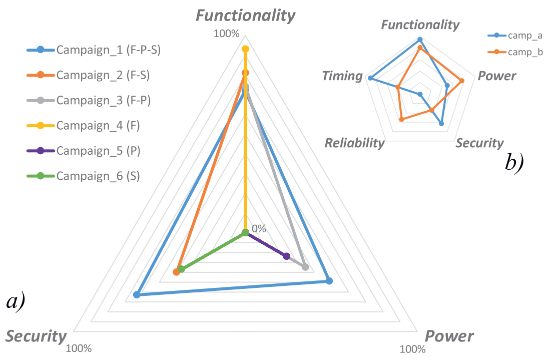


Fig. 2. Multidimensional verification campaigns (Radar-chart n-dimensional visualization).

Unfortunately, there is no established hardware design methodology supporting multidimensional verification plans for mutually influencing functional and extra-functional aspects. There is a very limited number of research works going beyond analysis of one extra-functional verification aspect under constraints of another as the complexity of the problem grows extremely fast with the number of dimensions (interdependent constraints) and the electronic system size. The first works in this direction are, for example, Vinco et al. [46] and Vain et al. [36].

Ultimately, results of multidimensional verification campaigns proposed in this work are subject to be represented in a multi-dimensional space, as illustrated in Fig. 2a. Here is shown an illustration of six hypothetical independent verification campaigns in a three-dimensional verification space. A verification campaign in this example shows the level of confidence in the different dimensions - (F)unctionality, (P)ower and (S)ecurity. In this illustrative example, only three aspects are taken into consideration. Obviously, on the demand the verification engineers can involve different dimensions. Here, the different colors of the lines represent different multi-dimensional spaces e.g. as Campaign_1 in blue lines stand for the verification result considering three extra functional aspects i.e., functional, power and security aspects at the same time. The figure shows the interdependency of these three requirements and thus can help the designers to choose the most suitable design combination. Subsequently, Compaign_2 represents the combination of functional and security aspects, Compaign_3 demonstrates the combination of functional and power aspect, etc. Thus the *Radar-charts, as shown in* Fig. 2b, are an instrument for summarizing multidimensional verification results for a large number of dimensions, (where the dimensions can be ordered to emphasize correlation or interdependencies between adjacent dimensions).

4.1. Motivational example

Single-dimension verification campaigns ignoring interdependencies between the dimensions may lead to gaps in the overall electronic system quality. As an example to show the importance of multidimensional verification, let us consider an actual verification campaign of an open-source NoC framework Bonfire [71,72].

```
process(write_en, write_pointer) begin --write pointer bug
  if write_en = '1' then
  write_pointer_in <= write_pointer(0)&write_pointer(3 downto 1);          -- Bug f1!
  else
    write_pointer_in <= write_pointer;
  end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer bug
  if (read_en = '1' and empty = '0') then
    read_pointer_in <= read_pointer(0)&read_pointer(3 downto 1);           -- Bug f1!
  else
    read_pointer_in <= read_pointer;
  end if;
end process;
```

```
process(write_en, write_pointer )begin --write pointer
  if write_en = '1' then
  write_pointer_in <= write_pointer(2 downto 0)&write_pointer(3);
  else
    write_pointer_in <= write_pointer;
end if;
end process;

process(read_en, empty, read_pointer) begin --read pointer
  if (read_en = '1' and empty = '0') then
    read_pointer_in <= read_pointer(2 downto 0)&read_pointer(3);
  else
    read_pointer_in <= read_pointer;
  end if;
end process;
```

Fig. 3. Bug f1 and its correction.

```
process(Healthy_packet, reset_counters, healthy_counter_out) begin
  if reset_counters  = '1' then
      healthy_counter_in <=  (others => '0');
  elsif Healthy_packet = '1' then                  -- Bug p1!
      healthy_counter_in <= healthy_counter_out + 1;
  else
      healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

```
process(Healthy_packet, reset_counters, healthy_counter_out,  faulty_counter_out) begin
  if reset_counters  = '1' then
      healthy_counter_in <=  (others => '0');
  elsif Healthy_packet = '1' and faulty_counter_out /= std_logic_vector(to_unsigned(0, faulty_counter_out'length)) then
      healthy_counter_in <= healthy_counter_out + 1;
  else
      healthy_counter_in <= healthy_counter_out;
  end if;
end process;
```

Fig. 4. Bug p1 and its correction.

The design under verification is in RTL VHDL and implements a 2×2 NoC infrastructure (processing elements excluded). The verification plan considered 2-dimensional verification campaign targeting *functionality* and *power consumption* requirements. For the former, assertion-based functional verification by simulation was employed targeting statement, branch, condition and toggle coverage metrics and satisfaction of a set of temporal simple-subset PSL assertions. For the latter, a set of power targets were extracted for the targeted silicon implementation assuming a particular switching activity (set to 12 mW in this example).

Among documented design errors in the Bonfire project, the bug *f1*, as shown in Fig. 3, is an example of a functional misbehavior due to improper usage of write and read pointers in the FIFO. The figure represents the code errors in the red line and the corrected versions of the code lines in blue. The bug f1 and the bug p1 demonstrate the error in Figs. 3 and 4, respectively. The bug *p1* causes violations of specified power consumption targets because of unnecessary excessive use of a fault-tolerance structure related counter. The report of such a power consumption is described in Table 2. The power consumption is shown in the cell Total Power which is composed of the dynamic power, i.e. the Switching Power in the interconnects and the Internal Power in the logic cells, and the insignificant (for the target technology) static leakage power Leak Power. As summarized in the first row, for the bug f1 the Total Power is equal to 10.211 (consistence with the power consumption requirement). Similarly, in the third row, which represents the power consumption for the correct version of the code, the total power is equal to 10.184. This report prove that even if there is a bug (bug f1) in the code but still the power consumption requirement is met. In contrary, for the bug p1, even though there is no functional errors, the Total Power consumption is reported which is equal to 22.137. Thus the bug p1 results in a double power consumption compared to the correct implementation and violates the power targets in the specification. This fact prove that it is critical to know how and where the code should be modified in order to reduce the power consumption as well as maintain functional correctness. In general, the above simple motivation example demonstrates the challenge of interdependency of different aspects when requirements in more than one dimension are present.

5. Machine learning to tackle the challenges of multidimensional verification

As it can be seen in the previous sections, multidimensional verification is a complex multi-aspect optimization problem. Machine learning algorithms are known to be able to learn complex relationships and have been used for several optimization problems. Section 3.5 has shown that machine learning techniques were already successfully used for estimating several different single verification metrics. This suggests that machine learning can be also used for solving multidimensional verification problem. There-

Table 2
Power consumption of the Bonfire system implementation: corrected and with bugs f1 and p1.

Bonfire system Implementation	Switching Power (mW)	Internal Power (mW)	Leak Power (pW)	Total Power (mW)
with *f1* bug	0.783	9.427	7.50e+05	10.211
with *p1* bug	0.757	21.379	6.93e+05	22.137
corrected	0.666	9.518	7.43e+05	10.184

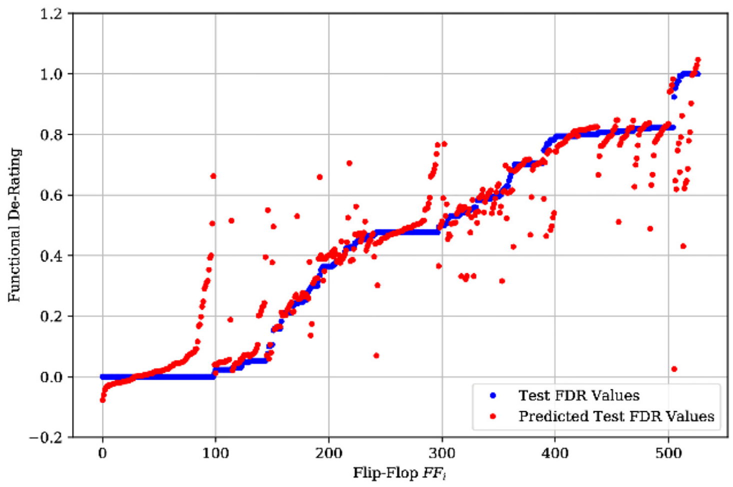


Fig. 5. Prediction of Functional De-Rating factors of the test data set by using a Support Vector Machine regression model (Training Size = 50%, Coefficient of Determination $R^2 = 0.844$).

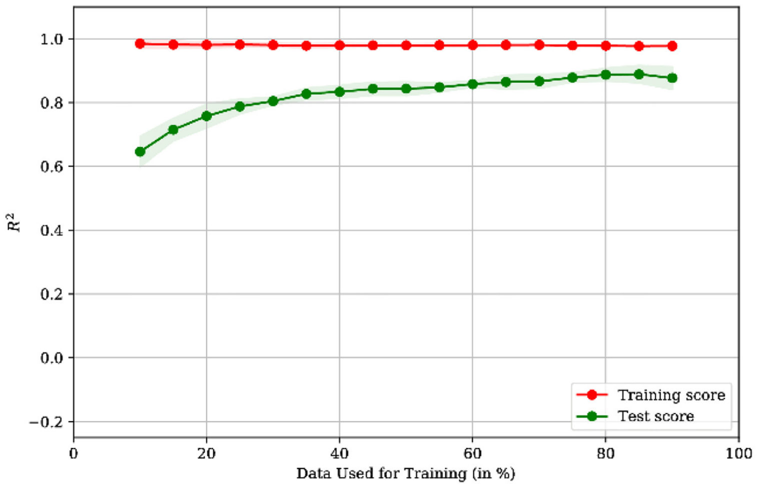


Fig. 6. Learning Curve for the Functional De-Rating prediction by using a Support Vector Machine regression model with different training sizes.

fore, an initial approach is proposed which is based on machine learning techniques in order to tackle this multidimensional verification challenge.

5.1. Proposed methodology

The proposed approach targets to predict two different verification metrics based on the same feature set extracted from the gate-level netlist of a given circuit. These two different metrics are Prediction of De-Rating and Path delay. The first metric to predict is the De-Rating or Vulnerability Factor, which are related to the reliability verification flow and a major metric of the failure analysis. The second metric is the path delay and related to the timing analysis. This metric is usually obtained during the synthesis or place and route stage of the design development. Therefore, machine learning can help to shift the analysis to an earlier design stage.

A possible application scenario consists in extracting a list of circuit feature and training a ML tool with a limited set of reference inputs (the values of the selected circuit features) and

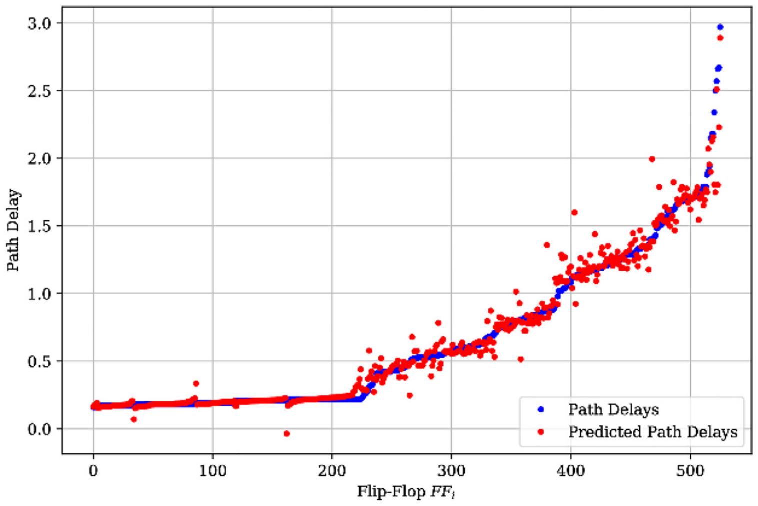


Fig. 7. Prediction of Path Delays of the test data set by using a Support Vector Machine regression model (Training Size = 50%, Coefficient of Determination $R^2 = 0.975$).

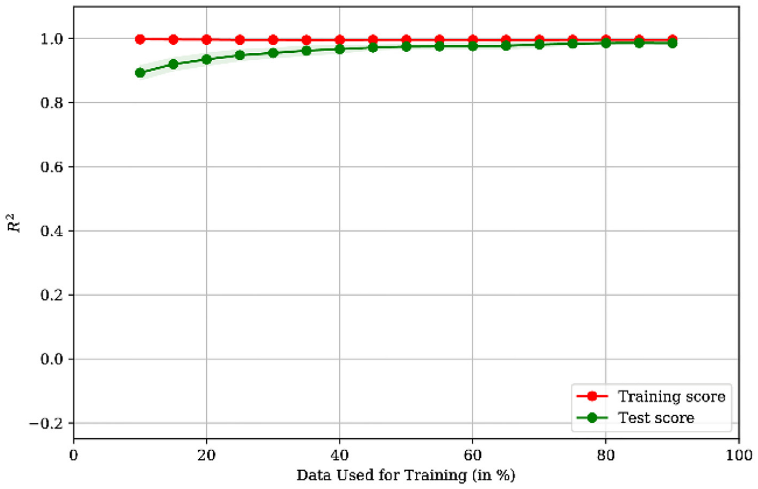


Fig. 8. Learning Curve for the Path Delay prediction by using a Support Vector Machine regression model with different training sizes.

expected outputs (reliability and timing metrics). Depending on the exhaustiveness of the training campaign, the trained ML tool can provide actual reliability metrics from a limited list of circuit features while spending far less resources (CPU time, EDA tools licenses, man-power) than using classical methods.

5.2. Prediction results

The proposed idea was implemented and evaluated on a practical example. Therefore, a set of features is defined which characterizes each flip-flop instance in the circuit. The feature set is composed of static elements (cell properties, circuit structure, synthesis attributes) and dynamic elements (signal activity). After extracting the features for the full list of circuit instances, reference data was obtained. The Functional De-Rating per flip-flop was determined through first-principles fault simulation approaches and the path delay was extracted by a classical static timing analysis. One part of the reference dataset is used to train the machine learning model and the remaining data is used to validate and benchmark the accuracy of the trained tool.

As a circuit under test, the Ethernet 10GE MAC Core was used which is available as RTL description from OpenCores. The circuit consists of control logic, state machines, FIFO controllers and memory interfaces. By synthesizing the design with NanGate FreePDK45 Open Cell Library, 1054 flip-flops have been identified and the corresponding features have been extracted.

Several machine learning models have been evaluated, such as the Linear Least Squares, Ridge (with linear and non-kernels), k-Nearest Neighbors (k-NN), Decision Tree (CART) and Support Vector regression (SVR, with linear and non-linear kernels). It has been noted that especially the linear models are not very suitable to predict the reliability metrics. The non-linear models perform much better and the Support Vector regression with Radial Basis Function (RBF) as kernel functions is among the best. Therefore, the SVR model with RBF kernel function is used for the following presentation of the prediction results. Figs. 5 and 7 show the prediction of the two metrics. When 50% (527 flip-flops) of the data are used to train the model and the remaining 50% was used to evaluate the model. The performance of regression models is usually evaluated by using the Coefficient of Determination (R^2) score and the model reaches a score of $R^2 = 0.844$ to predict the Functional De-Rating and $R^2 = 0.975$ to predict the path delay. Figs. 6 and 8 show the learning curve of the model. This curve describes the performance of the model for different sizes of the data set used for training and the remaining data set used for the evaluation. The learning curves suggest that by using more than 50% of the available data for training doesn't significantly improve the prediction performance. However, it can also be seen that by using less than 50% still a valuable prediction can be performed. Thus, by allowing a slight reduction of accuracy, the cost of an exhaustive analysis can still be reduced.

The practical example has shown that machine learning can be successfully applied for different verification purposes. In order use ML to support the multidimensional verification problem, features from different design stages need to be extracted and used to train a unified model or several separated models. These can be used to predict the required verification metrics.

6. Conclusion

In the recent years, numerous extra-functional aspects of electronic systems were brought to the front and imply verification of hardware design models in multidimensional space along with the functional concerns of the target system. Targeting at understanding of this new verification paradigm, we have performed a comprehensive analysis of the state of the art and presented a taxonomy for multidimensional hardware verification aspects, an up-to-date survey of related research works and trends towards enabling the multidimensional verification concept and investigated the potential of machine learning based techniques to support the concept. As the result of the performed analysis of the state of the art we have outlined a gap in methodologies and tools for holistic multidimensional verification of hardware design models and the key challenges.

Declaration of competing interest

All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.

Acknowledgments

This research was supported in part by projects H2020 MSCA ITN RESCUE funded from the EU H2020 programme under the MSC grant agreement no. 722325, by the Estonian Ministry of Education and Research institutional research grant no. IUT19-1 and by European Union through the European Structural and Regional Development Funds.

References

[1] I. Verbauwhede, Security adds an extra dimension to IC design: future IC design must focus on security in addition to low power and energy, in: IEEE Solid-State Circuits Magazine, 9, Fall 2017, pp. 41–45.
[2] W. Chen, S. Ray, J. Bhadra, M. Abadir, L.C. Wang, Challenges and trends in modern SoC design verification, in: IEEE Design & Test, 34, Oct. 2017, pp. 7–22.
[3] A. Piziali, Functional Verification Coverage Measurement and Analysis, Springer, 2008.
[4] S. Ullah, M. Iqbal, A.M. Khan, A survey on issues in non-functional requirements elicitation, in: Int. Conf. on Computer Networks and Information Technology, Abbottabad, 2011, pp. 333–340.
[5] L.M. Cysneiros, E. Yu, Non-Functional requirements elicitation, in: J.C.S. do Prado Leite, J.H. Doorn (Eds.), Perspectives On Software Requirements. The Springer International Series in Engineering and Computer Science, 753, Springer, Boston, MA, 2004.
[6] M.L. Fair, Reliability, availability, and serviceability (RAS) of the IBM eServer z990, IBM J. Res. Dev. 48 (3.4) (May 2004) 519–534.
[7] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, Non-Functional requirements, Software Engineering, Kluwer Academic, 2000.
[8] P. Singh, A.K. Tripathi, Exploring problems and solutions in estimating testing effort for non functional requirement, Int. J. Comput. Technol. 3 (2b) (2012) 284–290.
[9] E.R. Poort, N. Martens, I. Van de Weerd, H. Van Vliet, How architects see non-functional requirements: beware of modifiability, in: Requirements Engineering: Foundation for Software Quality, Springer, Berlin Heidelberg, 2012, pp. 37–51.
[10] D. Ameller, C. Ayala, J. Cabot, X. Franch, How do software architects consider non-functional requirements: an exploratory study, in: Requirements Engineering Conference (RE), 2012, pp. 41–50.
[11] M. Glinz, On non-functional requirements, in: Requirements Engineering Conference, 2007. RE'07, IEEE, 2007, pp. 21–26.
[12] L. Motus, Analytical study of quantitative timing properties of software, 5th EUROMICRO Workshop on Real-Time Systems, 1993.
[13] M. Jenihhin, G. Squillero, T.S. Copetti, V. Tihhomirov, S. Kostin, M. Gaudesi, F. Vargas, J. Raik, M. Sonza Reorda, L. Bolzani Poehls, R. Ubar, G.C. Medeiros, Identification and rejuvenation of NBTI-Critical logic paths in nanoscale circuits, JETTA 32 (3) (June 2016) 273–289.
[14] J. Bhadra, M.S. Abadir, L.C. Wang, S. Ray, A survey of hybrid techniques for functional verification, in: IEEE Design & Test of Computers, 24, 2007, pp. 112–122.
[15] S. Mukherjee, Architecture design for soft errors, Morgan Kauf (2008).
[16] A. Ptzmann, M. Hansen, Anonymity unlinkability undetectability unobservability pseudonymity and identity management, A Consolidated Proposal for Terminology version 0.31, 2008.
[17] R. Ubar, et al., Diagnostic modeling of digital systems with multi-level DDs, in: R. Ubar, J. Raik, Vierhaus H.Th. (Eds.), Design and Test Technology For Dependable, SoC, 2011, pp. 92–118.
[18] V. Guarnieri, Mutation analysis for SystemC designs at TLM, in: 2011 12th Latin American Test Workshop (LATW), Porto de Galinhas, 2011, pp. 1–6.
[19] J. Hansson, B. Lewis, J. Hugues, L. Wrage, P. Feiler, J. Morley, Model-Based verification of security and non-functional behavior using AADL, IEEE Secur. Priv. (2009) 1–1.
[20] J. Sepulveda, D. Aboul-Hassan, G. Sigl, B. Becker, M. Sauer, Towards the formal verification of security properties of a Network-on-Chip router, in: 2018 IEEE 23rd European Test Symposium (ETS), Bremen, 2018, pp. 1–6, doi:10.1109/ETS.2018.8400692.
[21] T. Boraten, D. DiTomaso, A.K. Kodi, Secure model checkers for Network-on-Chip (NoC) architectures, in: 2016 Int. Great Lakes Symposium on VLSI (GLSVLSI), Boston, MA, 2016, pp. 45–50.
[22] H.M.G. Wassel, Networks on chip with provable security properties, IEEE Micro. 34 (3) (May-June 2014) 57–68.
[23] M.A. Kochte, M. Sauer, L.R. Gomez, P. Raiola, B. Becker, H.J. Wunderlich, Specification and verification of security in reconfigurable scan networks, in: 2017 22nd IEEE European Test Symposium (ETS), Limassol, 2017, pp. 1–6.
[24] L.W. Kim, J.D. Villasenor, Dynamic function verification for system on chip security against hardware-based attacks, IEEE Transa. Reliab. 64 (4) (Dec. 2015) 1229–1242.
[25] Hu Wei, et al., Imprecise security: quality and complexity tradeoffs for hardware information flow tracking, in: IEEE/ACM Int. Conference on Computer-Aided Design (ICCAD), Austin, TX, 2016, pp. 1–8.
[26] A. Nahiyan, M. Sadi, R. Vittal, G. Contreras, D. Forte, M. Tehranipoor, Hardware trojan detection through information flow security verification, in: 2017 IEEE International Test Conference (ITC), Fort Worth, TX, 2017, pp. 1–10.
[27] M.A. Kochte, R. Baranowski, M. Sauer, B. Becker, H.J. Wunderlich, Formal verification of secure reconfigurable scan network infrastructure, in: 2016 21th IEEE European Test Symposium (ETS), Amsterdam, 2016, pp. 1–6.
[28] M. Rocchetto, N.O. Tippenhauer, Towards formal security analysis of industrial control systems, in: ACMA sia Conf. Comput. Commun. Secur., 2017, pp. 114–126.
[29] M. Yoshimura, T. Bouyashiki, T. Hosokawa, A hardware trojan circuit detection method using activation sequence generations, in: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), Christchurch, 2017, pp. 221–222.
[30] F.K. Lodhi, S.R. Hasan, O. Hasan, F. Awwad, Formal analysis of macro synchronous micro asychronous pipeline for hardware trojan detection, in: NORCAS, Oslo, 2015, pp. 1–4.
[31] S. Deng, W. Xiong, J. Szefer, Cache timing side-channel vulnerability checking with computation tree logic, in: Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, ser. HASP '18, New York, NY, USA, ACM, 2018, p. 2. 1–2:8.
[32] T. Zhang, R.B. Lee, New models of cache architectures characterizing information leakage from cache side channels, in: Proceedings of the 30th Annual Computer Security Applications Conference, ser. ACSAC'14, New York, NY, USA, ACM, 2014, pp. 96–105.
[33] P. Canones, B. Köpf, J. Reineke, Security analysis of cache replacement policies, CoRR, 2017 arXiv:1701.06481.

[34] Z. Hanna, Verifying security aspects of SoC designs with Jasper app, (white paper), Jasper Design Automation (Cadence), 2013.
[35] JasperGold Security Path Verification App, Cadence, http://www.cadence.com.
[36] J. Vain, A. Kaur, L. Tsiopoulos, J. Raik, M. Jenihhin, Multi-view modeling for MPSoC design aspects, in: 2018 16th Biennial Baltic Electronics Conference (BEC), Tallinn, 2018, pp. 1–6.
[37] Q. Ge, Y. Yarom, D. Cock, G. Heiser, A survey of microarchitectural timing attacks and countermeasures on contemporary hardware, J. Cryptogr. Eng. 8 (1) (2018) 1–27.
[38] Y. Lyu, P. Mishra, A survey of side-channel attacks on caches and countermeasures, J. Hardw. Syst. Secur. 2 (1) (Mar 2018) 33–50.
[39] A. Savino, S. Di Carlo, A. Vallero, G. Politano, D. Gizopoulos, A. Evans, RIIF-2: toward the next generation reliability information interchange format, IEEE IOLTS (2016) 173–178.
[40] C. Liu, J. Layland, Scheduling algorithms for multi programming in a hard real time environment, J. ACM 20 (1973) 46–61.
[41] F.S. Gonçalves, D. Pereira, E. Tovar, L.B. Becker, Formal verification of AADL models using UPPAAL, in: 2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC), Curitiba, 2017, pp. 117–124.
[42] D. Burlyaev, P. Fradet, Formal verification of automatic circuit transformations for fault-tolerance, in: 2015 Formal Methods in Computer-Aided Design (FMCAD), Austin, TX, 2015, pp. 41–48.
[43] B.W. Thompto, B. Hoppe, Verification for fault tolerance of the IBM system z microprocessor, in: Design Automation Conference, Anaheim, CA, 2010, pp. 525–530.
[44] S. Kan, M. Lam, T. Porter, J. Dworak, A case Study: pre-Silicon SoC RAS validation for NoC server processor, in: MTV, 2016, pp. 19–24.
[45] S. Vinco, M. Lora, E. Macii, M. Poncino, IP-XACT for smart systems design: extensions for the integration of functional and extra-functional models, in: 2016 Forum on Specification and Design Languages (FDL), Bremen, 2016, pp. 1–8.
[46] S. Vinco, Y. Chen, F. Fummi, E. Macii, M. Poncino, A layered methodology for the simulation of extra-functional properties in smart systems, in: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 36, 2017, pp. 1702–1715.
[47] G. Aleksandrowicz, et al., Designing reliable cyber-physical systems, in: F. Fummi, R. Wille (Eds.), Languages, Design Methods, and Tools for Electronic System Design. Lecture Notes in Electrical Engineering, 454, Springer, Cham, 2018.
[48] Eli Arbel, Shlomit Koyfman, Prabhakar Kudva, Shiri Moran, Automated detection and verification of parity-protected memory elements, in: Proc. IEEE/ACM ICCAD, 2014, pp. 1–8.
[49] M. Maniatakos, Y. Makris, Workload-driven selective hardening of control state elements in modern microprocessors, in: VTS, 2010, pp. 159–164.
[50] S. Ganapathy, R. Canal, D. Alexandrescu, E. Costenaro, A. González, A. Rubio, INFORMER: an integrated framework for early-stage memory robustness analysis, in: 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014, pp. 1–4.
[51] S. Avramenko, S.P. Azad, S. Esposito, B. Niazmand, M. Violante, J. Raik, M. Jenihhin, QoSinNoC: analysis of qos-Aware NoC architectures for mixed-criticality applications, in: 21st IEEE Int. Symp. DDECS, 2018, pp. 1–6.
[52] S. Avramenko, Upgrading QoSinNoC: efficient routing for mixed-criticality applications and power analysis, in: IEEE VLSI-SoC, Verona, 2018, pp. 1–6.
[53] S. Rubini, F. Singhoff, J. Hugues, Modeling and verification of memory architectures with AADL and REAL, in: 2011 16th IEEE International Conference on Engineering of Complex Computer Systems, Las Vegas, NV, 2011, pp. 338–343.
[54] H. Wang, X. Zhou, Y. Dong, L. Tang, A hierarchical verification procedure of timed petri-net model for real-time embedded systems, in: 2010 2nd International Conference on Information Engineering and Computer Science, Wuhan, 2010, pp. 1–4.
[55] H. Wang, X. Zhou, Y. Dong, L. Tang, Timing properties analysis of real-time embedded systems with AADL model using model check, in: IEEE Int. Conf. on Progress in Informatics and Computing (PIC), 2010, pp. 1019–1023.
[56] A. Rafiev, F. Xia, A. Iliasov, A. Romanovsky, A. Yakovlev, Selective abstraction for estimating extra-functional properties in Networks-on-Chips using archon framework, in: 2017 17th International Conference on Application of Concurrency to System Design (ACSD), Zaragoza, 2017, pp. 80–85.
[57] P. Khondkar, Low-Power Design and Power-Aware Verification, Springer, 2018.
[58] D. Lorenz, Non-invasive power simulation at system-level with systemc, PATMOS 2012. LNCS (7606), Springer, 2012.
[59] S. Orcioni, et al., Energy estimation in SystemC with powersim, Integr. VLSI J. (55) (2016) 118–128.
[60] ANSI/IEEE 1801-2015 - IEEE Standard for design and verification of Low-Power, energy-aware electronic systems, March 2016,
[61] Si2 Common Power Format, v2.1, Silicon Integration Initiative, 2014.
[62] E.Y. Kang, D. Mu, L. Huang, Q. Lan, Verification and validation of a cyber-physical system in the automotive domain, in: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, 2017, pp. 326–333.
[63] A. Danese, G. Pravadelli, I. Zandonà, Automatic generation of power state machines through dynamic mining of temporal assertions, in: 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016, pp. 606–611.
[64] J. Zimmermann, S. Stattelmann, A. Viehl, O. Bringmann, W. Rosenstiel, Model-driven virtual prototyping for real-time simulation of distributed embedded systems, in: 7th IEEE Int. Symposium on Industrial Embedded Systems (SIES'12), Karlsruhe, 2012, pp. 201–210.
[65] R. Görgen, et al., CONTREX: design of embedded mixed-criticality CONTRol systems under consideration of EXtra-Functional properties, in: 2016 Euromicro Conference on Digital System Design (DSD), Limassol, 2016, pp. 286–293.
[66] A.W. Ruan, Y.B. Liao, P. Li, W.C. Li, W. Li, Throughput estimation for modelsim simulator tool based HW/SW co-verification system, in: 2009 International Conference on Communications, Circuits and Systems, Milpitas, CA, 2009, pp. 1014–1018.
[67] M. Khamis, S. El-Ashry, A. Shalaby, M. AbdElsalam, M.W. El-Kharashi, A configurable RISC-V for noc-Based MPSoCs: a framework for hardware emulation, in: 2018 11th International Workshop on Network on Chip Architectures (NoCArc), Fukuoka, 2018, pp. 1–6.
[68] JasperGold Connectivity Verification App, Cadence, http://www.cadence.com.
[69] S.K. Roy, Top level SOC interconnectivity verification using formal techniques, in: The 8th Int. Workshop on Microprocessor Test and Verification, Austin, TX, USA, 2008, pp. 63–70.
[70] M. Elver, V. Nagarajan, McVerSi: a test generation framework for fast memory consistency verification in simulation, in: 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), Barcelona, 2016, pp. 618–630.
[71] S.-P. Azad, B. Niazmand, K. Janson, J. Raik, Github Bonfire Project (2017), https://github.com/Project-Bonfire/ (accessed 1 June 2019).
[72] S.P. Azad, From online fault detection to fault management in Network-on-Chips: a ground-up approach, in: 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Dresden, 2017, pp. 48–53.
[73] S.-C. Fang, C.-C. Weng, C.-K. Tseng, C.-W. Hsu, J.-L. Liao, S.-Y. Huang, C.-L. Lung, D.-M. Kwai, SoC power analysis framework and its application to power-thermal co-simulation, in: 2011 Int. Symp. on VLSI Design, Automation and Test, April 2011, pp. 1–4.
[74] G. Vece, M. Conti, S. Orcioni, Transaction-level power analysis of VLSI digital systems, Integr. VLSI J. 50 (2015) 116–126.
[75] M. Giammarini, M. Conti, S. Orcioni, System-level energy estimation with powersim, in: 2011 18th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS), December 2011, pp. 723–726.
[76] Bell Labs, Verifying Multi-threaded Software with Spin, (1980). http://spinroot.com/ (accessed 1 June 2019).
[77] SMT Steering Committee, The International Satisfiability Modulo Theories (SMT) Competition. http://www.smtcomp.org/ (accessed 1 June 2019).
[78] Carnegie Mellon University, Architecture Analysis and Design Language. http://www.aadl.info/aadllcurrentsite/ (accessed 1 June 2019).
[79] S. Dai, Y. Zhou, H. Zhang, E. Ustun, E.F.Y. Young, Z. Zhang, Fast and accurate estimation of quality of results in high-level synthesis with machine learning, in: 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018, pp. 129–132.
[80] K. Hasegawa, M. Oya, M. Yanagisawa, N. Togawa, Hardware Trojans classification for gate-level netlists based on machine learning, in: 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), 2016, pp. 203–206.
[81] K. Hasegawa, M. Yanagisawa, N. Togawa, Hardware Trojans classification for gate-level netlists using multi-layer neural networks, in: 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS), 2017, pp. 227–232.
[82] E. Alpaydin, F. Bach, Introduction to Machine Learning, MIT Press, 2014.
[83] B. Yu, D.Z. Pan, T. Matsunawa, X. Zeng, Machine learning and pattern matching in physical design, in: The 20th Asia and South Pacific Design Automation Conference, 2015, pp. 286–293.
[84] B. Li, P.D. Franzon, Machine learning in physical design, in: 2016 IEEE 25th Conference on Electrical Performance Of Electronic Packaging And Systems (EPEPS), 2016, pp. 147–150.
[85] L. Bai, L. Chen, Machine-Learning-Based early-stage timing prediction in SoC physical design, in: 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), 2018, pp. 1–3.
[86] K. Hasegawa, M. Yanagisawa, N. Togawa, Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier, in: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1–4.

Xinhui Lai is one of the early stages researchers in the RESCUE European Training Network. She is doing her research work and Ph.D in Tallinn University of Technology which is one of the institutions evolved in the RESCUE project. She got her bachelor and master diploma in Electronic Engineering from Politecnico di Torino, Italy, in October 2014 and April 2017 respectively. Her research is focused on design error functional verification and automated debug, i.e. localization and correction, as well as verification of extra-functional interdependent aspects in nanoelectronic system design such as security, reliability, power/performance envelope.

Aneesh Balakrishnan is an Early Stage Researcher in the RESCUE European Training Network. Within the project, he is a Research and Development Engineer at iRoC Technologies, France and also a Ph.D. Candidate in the department of computer systems at Tallinn University of Technology, Estonia. Aneesh has a master degree in Communication and Multimedia Engineering from Friedrich-Alexander University, Erlangen-Nurnberg, Germany in July 2016 and holds a Bachelor of Engineering in Electronics and Communication Engineering from Anna University of Technology, India. His current research will address today's high-performance designs requirements in term of validation and reliability. The objective of the research is to significantly enhance and develop new statistical, probabilistic methods and algorithms.

Thomas Lange is an Early Stage Researcher in the RESCUE European Training Network. Within the project he is a Research and Development Engineer at iRoC Technologies, France and also Ph.D. Candidate in Computer and Systems Engineering at Politecnico di Torino, Italy. Thomas holds a Bachelor's and Master's degree in Computer Engineering from Technische Universität Berlin. In his research he is investigating the effects of transient faults for high reliability applications in harsh environments. His main interest includes the development of new models and assessment techniques for transient faults, as well as new mitigation and error management techniques with the focus on hardware capabilities.

Maksim Jenihhin is a professor of Computing Systems Reliability at the Department of Computer Systems of Tallinn UT. He holds M.*Sc*. and Ph.D. degrees in Computer Engineering from Tallinn UT (2004 and 2008 respectively). His research interests include methodologies and EDA tools for hardware design, verification and debug as well as nanoelectronics reliability and manufacturing test topics. Maksim is a project coordinator for H2020 RESCUE - Interdependent Challenges of Reliability, Security and Quality in Nanoelectronic Systems Design.

Tara Ghasempouri is a Postdoctoral Researcher at Tallinn University of Technology in Computer Systems department. Her group is mainly focused on three categories such as fault tolerance, verification and safety/security of systems. She is interested in finding innovative solutions for the verification process at the different level of abstraction. Her research topic is also focused on Hardware Security. She received a Ph.D. degree in Computer Science from University of Verona, Italy. During her Ph.D. program, she has researched on different kind of verifications and specially Assertion-based Verification on the embedded system. She is a member of IEEE Computer Society and she was a reviewer for different conferences such as ETS, VLSI-SOC and etc.

Jaan Raik is a professor of digital systems verification at the Department of Computer Systems of Tallinn University of Technology and the leader of the Center for Dependable Computing Systems Design (DCSD). Prof. Raik received his M.*Sc*. and Ph.D. degrees in Computer Engineering from Tallinn University of Technology in 1997 and in 2001, respectively. He is a member of IEEE Computer Society, HiPEAC and a member of steering/program committees of several conferences. He has co-authored more than 200 scientific publications.

Dan Alexandrescu (IEEE Member'07-Senior Member'13) is the CEO of IROC Technologies. Dan holds a M. Eng. in Electronics from Politehnica Bucharest, Romania, a M.A.S. in Microelectronics from Joseph Fourier University, Grenoble, France and a Ph.D in Microelectronics from INPG, Grenoble Institute of Technology, France. He specializes in the design, optimization and improvement of highly-reliable microelectronic circuits. He contributed to the organization of reliability-centric workshops and symposia (Program Co-Chair for multiple IOLTS editions, Finance and General Co-Chair for several SELSE editions) and he prepared many publications in the field of reliability and radiation-induced effects.

Appendix 3

III

X. Lai, M. Jenihhin, J. Raik, and K. Paul, “Pascal: Timing sca resistant design and verification flow,” in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pp. 239–242, IEEE, 2019

PASCAL: Timing SCA Resistant Design and Verification Flow

Xinhui Lai[1], Maksim Jenihhin[1], Jaan Raik[1], Kolin Paul[1,2]
[1] Computer Systems, Tallinn University of Technology, Estonia
[2] Department of Computer Science & Engg. Indian Institute of Technology Delhi, India
email:{Xinhui.Lai,Maksim.Jenihhin,Jaan.Raik,Kolin.Paul}@taltech.ee

***Abstract*—A large number of crypto accelerators are being deployed with the widespread adoption of IoT. It is vitally important that these accelerators and other security hardware IPs are *provably secure*. Security is an extra functional requirement and hence many security verification tools are not mature. We propose an approach/flow – PASCAL – that works on RTL designs and discovers potential Timing Side Channel Attack (SCA) vulnerabilities in them. Based on information flow analysis, this is able to identify Timing Disparate Security Paths that could lead to information leakage. This flow also (automatically) eliminates the information leakage caused by the timing channel. The insertion of a lightweight *Compensator Block* as balancing or compliance FSM removes the timing channel with minimum modifications to the design with no impact on the clock cycle time or combinational delay of the critical path in the circuit.**

I. Introduction

Security is not a first class citizen in (hardware) design and is rarely considered during design space exploration. Bugs or vulnerabilities can originate from design flaws, some of which can be fully eliminated after a complete verification. The goal of the adversary in a security critical application, is to learn information that one has no legitimate access to, e.g. the classified data or secret keys. Novel attack vectors like side-channel analysis rely on design features, to build efficient exploits that undermine assumptions regarding the accessibility of internal secret information in a computing system. For example, Timing Driven Attacks exploit timing differences in execution traces as the information flow is via different paths with the same start and end nodes (controllable and observable) to derive the secret information.

Denning et. al. introduced the concept of secure information flow in a computer system whereby it can be shown that no unauthorized flow of information is possible due to control and data flow [1]. However, in recent years, side channels or out of band data channels have been exploited to exfiltrate or deduce secret information. Consequently, Information Flow Tracking (IFT) has evolved and has been used as a formal methodology for modeling and reasoning about security properties related to integrity, confidentiality of side channels. The problem becomes more interesting and hard because high-level architecture abstractions are translated into transparent micro architecture implementations. While the hardware behavior in the micro-architecture can cause additional information flows which can be gainfully exploited to form these side channels.

As opposed to physical Side Channels Attacks (SCA) like differential power attacks etc. that require physical access to the computer system, Timing SCA can be launched (relatively) easily on general purpose compute environments that contain a memory hierarchy or performance enhancing microarchitecture features like speculative execution. The key invariant in these attacks is that there are different timing paths that provide out of band information. Security path verification addresses a specific, important aspect of overall security verification by checking access to the secure data on the hardware to make certain that attackers access the secure (secret) data through illegal logic paths. For example, in Figure 1, there are paths

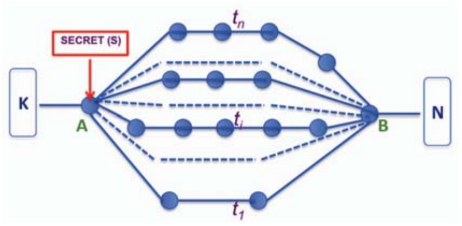


Fig. 1: Timing Disparate Paths

from A to B which is controlled by the node S containing the secret. Tools do Taint Propagation/Taint Analysis, which is a conservative approximation of secure information flow analysis, to find such paths [2]. A timing side channel exists, if the contents of S can be derived/deduced by analyzing time of arrival of K at N. We call these two or more paths with unequal transit time as **Timing Disparate Security Paths**. And these Timing Disparate Security Paths will be potentially vulnerabilities open for a timing side channel exploit.

The primary contribution of our work is a secure automated digital design flow – **PA**th based **S**ide **C**hannel **A**na**L**ysis (**PASCAL**) – that creates a secure IP core or system-on-chip. The proposed flow starts from the RTL design and the threat model and uses a state of the art Security Path verification tool to identify potential timing side channel vulnerabilities and proposes a method to remove them by enforcing uniform timing to remove data dependent instruction cycle count variations in the timing side channels.

The remainder of the paper is organized as follows. Section II summarizes the state of the art in this area. The next section details the approach used and presents the key algorithmic contribution in this work for identifying Timing Disparate

Security Paths while Section IV presents a lightweight method for Timing SCA Resistant Design using the results of the method proposed in the previous section. Section V describes the implementation results on a widely used crypto core and also demonstrates the efficacy of the proposed mitigation method. Section VI summarises the contributions of the paper and provides directions for future work.

II. Related Work

Timing side channel attacks are known to be a hard and a very important problem in modern systems. They have been used to extract cryptographic secrets from running systems.Even differential privacy systems are not immune to these attacks. And these are possible using both remoteand local adversaries. Koeune et. al present an indepth tutorial on Side Channel Attacks [3].

A popular approach for defending against both local and remote timing attacks is to ensure that the low-level instruction sequence does not contain instructions whose performance depends on secret information. This can be enforced by manually re-writing the code, as was done in OpenSSL or by changing the compiler to ensure that the generated code has this property [4].

While methods for high performance design or low power are available, design for security is still adhoc. Only recently, systematic methods support design for trust and security have been described in literature [5].Menichelli et. al present an exploration approach centered on high level simulation based on SystemC to suggest improvements in the knowledge and identification of the weaknesses in cryptographic algorithm implementations [6]. Ardeshiricham et. al. have proposed an information flow based method for secure hardware design [7] by analyzing all logical code flows of the RTL code. In contrast VeriCoq-IFT converts designs from HDL to Coq to analyze a formal security properties [8]. SecVerilog requires explicit annotating each variable in the design with a security label — this is similar to using a type system to track information flow in the code [9]. Deng et. al. have proposed a Computation Tree Logic to model execution paths of the processor cache logic and derive formulas for paths that can lead to timing side-channel vulnerabilities [10].

Most of the mitigation techniques that have been proposed try to remove data dependent instruction cycle count variations by balancing timing or do a power flattening to remove power peaks/anomalies [11]. In some cases, Pipeline randomization for power and timing is also attempted. Alternatively, packet route randomisation as a mechanism to increase NoC resilience has also been proposed [12]. Recently, Jiang et. al. have proposed a high-level synthesis (HLS) infrastructure that incorporates static information flow analysis to remove timing channels in a verifiable manner on HLS-generated hardware accelerators [13].

The methodology proposed in this paper is based on a formal method that can identify **all** Timing Disparate Security Paths at RT level and improve the state of the art is a simple mitigation scheme for potential SCA vulnerable timing channels.

In the next section, we discuss the proposed method for discovering Timing Disparate Security Paths in RTL designs.

III. Methodology

Hardware implementations of encryption algorithms are being increasingly used as hardware is regarded as more effective root of trust. RSA is a asymmetric cryptographic algorithm and has been shown to be vulnerable to Timing SCAs and mitigation techniques have also been proposed. However, the major focus continues to rely on verifying the correctness of encryption algorithms and their implementation in software and hardware. We present an approach based on RT level analysis that allows a precise understanding of possible flows for side channels based on timing. The methodology relies on a formal analysis tool Cadence JasperGold Security Path Verification App (JG SPV) [14]. The original objective of the tool is for security verification by checking access or leak of the secure data on the hardware to make certain that the attacker cannot breach the authentication logic and seek the secure data through illegal paths.

Based on a formal method of path sensitization from the secret information to the output observable points, we propose a method that can detect possible Timing-Disparate Paths in RTL designs which could be exploited as Timing Side Channel(s). As a result of this analysis, a simple and effective retiming of Timing SCA sensitive paths is proposed to make the design immune for the threats under the chosen threat model. We illustrate this on a standard RSA RTL Verilog code.

Algorithm 1: Example: RSA Modulus Code

```
Input: $C_m, P_n$; // C is the m bits cipher text, P is the n
       bits private key
Output: $O_m$; // O is the m bits output plain text
1 $R_0 \leftarrow Montgomery(C_m)$ and $R_1 \leftarrow Montgomery(1))$
2 $j \leftarrow 0$
3 while $j \leq n-1$ do
4     $R_0 \leftarrow Montgomery\_Reduction(R_1 * R_1)$
5     if $P[j]$ then
6         $R_0 \leftarrow Montgomery\_Reduction(R_0 * R_1)$
7     end
8     $O_m \leftarrow Montgomery^{-1}(R_0)$
9 end
```

The decryption of the RSA modulus in **Algorithm 1** uses Montgomery modular multiplication with square-and-multiply algorithm. Here we did not mention the details about how to choose the key or how the Montegomery algorithm works but focus on explaining the unintended timing channels in RSA which can be used by attacker to reverse the private key. In **Algorithm 1**, n, the bit number of P_n, decides total loop times while value of single bit of P_n: P[j] determine the operations for each single loop – only when P[j] equal to 1, statements at Line 5,6,7 will be executed while P[j] equal to 0 will not. For the decryption of RSA, the total operations need to be executed might be different with different private key duing to the above reasons. Assuming the time for single bit P[j] is

$t_{\mathbf{P[j]}}$, the final execution time will be $t_{total} = \sum_{j=0}^{n} t_{\mathbf{P[j]}}$. Thus keys with different number of '1s' will cause the different execution time. This will open a timing side channel for the attackers.

For this Timing SCAs, the **PASCAL** is shown in figure 2. Firstly, we use JG SPV to analyze if there is one or several paths, from a variable deemed to be secure and unobservable to the output, exist. JG SPV uses a special path sensitization technology implying taint analysis to find if private key **P** can be propagated to the output **O**. Then if the path exist, JG SPV will give a counterexample along with an execution trace detailing: the exact number of clock cycles(say X). As shown in the figure 3, the example shows waveforms of related signals along the path. We use the command "*[get_property_info -list{max_length} property_exponent_to_finish]*" to get the total execution time(clock based) of an exist path for this specified secure signal pair. Here it needs 44 clock cycles(additional 2 clock cycles are for setting up)to propagate. After that, JG SPV will be used to find another functional path (if it exists) from **P** to the output **O** with a time length different from X cycles. This is achieved by invoking JG SPV on a modified design, shown in Figure 4. A counter is added which drives the multiplexer to select the situation where the DUV both finish the decryption *AND* also the length of the execution trace Y is not equal to X. If JG SPV finds another path with an execution trace length not equal to X and Y, it is added to the *Union Clause* of the multiplexer select condition and the process is repeated until they find all the timing classes.

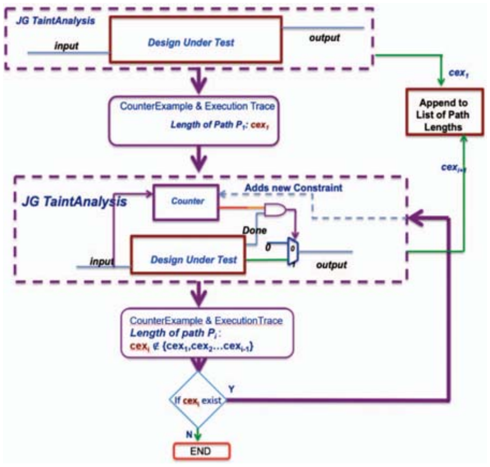


Fig. 2: **PASCAL**: Graphical Representation

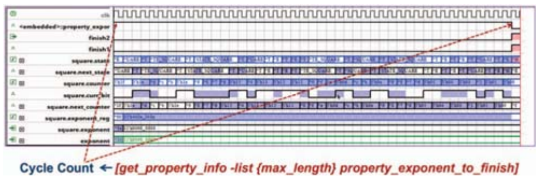

Cycle Count
[get_property_info -list {max_length} property_exponent_to_finish]

Fig. 3: Counter Example and Execution Trace

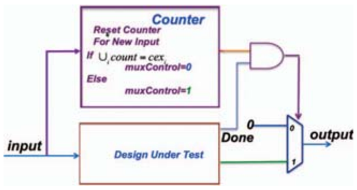


Fig. 4: **Modified DUV**

IV. Timing SCA Secure Design Flow

We also propose a method that aims to achieve timing-sensitive noninterference for the synthesized design, via which it is ensured that confidential or secret values cannot be revealed by the observing/measuring the timing of events at observable ports. An intuitive method to remove this Timing SCA vulnerability is to insert additional registers in the faster paths using path-balanced scheduling [15]. However, as shown in Figure 1, there could be many paths $t_1, t_2, \cdots, t_n$in the same basic block. Assume without loss of generality, that there are $n/2$ paths each differing by one cycle. Hence a path balanced scheduling synthesis procedure would insert $1 + 2 + 3 + \cdots + n/2$ or $\mathcal{O}(n^2)$ registers.

The method we propose is shown in **Algorithm 2**. Since Timing Disparate Security Paths result in a Timing SCA vulnerability only if they are observable at user interfaces (output ports), it can relax the constrains in the path balanced scheduling approach and enforce indistinguishable timing behaviors at the observable points in the design. Clearly, for the basic block or the core to be timing insensitive at the observable points, the output should be observable modulo t_{max} cycles where $t_{max} = maximum(t_1, t_2, \cdots, t_n)$. We enable the output port/interface every t_{max} cycles using a counter and an AND gate. This small additional circuitry acts as the **Compensator** or **balancing/compliance FSM** and provides the (read) enable / data ready signal for observable interface.

This therefore, leads to a very simple synthesis technique for ensuring a path balanced design with a single lightweight **Compensator Block** at the observable points of interest in the design. The additional circuit has a very small overhead counter which counts upto t_m to generate the control input for the AND gate which provides the enable signal to the observable register. The counter is reset every time a new input enters the basic block. This additional logic incurs no penalty in the critical path of the system and avoids resource duplication since it has a uniform counter where the results from the different Timing Disparate Security Paths are delivered to the observable interface with the same latency.

V. Results

The Montgomery modular multiplication with square-and-multiply algorithm based RSA cryptographic RTL implemen-

```
Algorithm 2: Timing Channel Removal
  Input: Design Under Verification (DUV)
  Output: Secure Design Under Test (sDUV)
1 $P \leftarrow$ PACSCAL(DUV) ;
2 $t_m$ = findMaxExecutionLength(P);
3 Ccompensator Logic Block $\leftarrow$ Counter($t_m$) + ResetLogic;
4 sDUV $\leftarrow$ Compensator Logic Block + DUV
```

tation is vulnerable to timing SCA. This is because for different keys the time differences are dependent on the number of '1s' in the key as explained in the Algorithm 1. In figure 5, the time required to generate the timing disparate classes for 32-bits RSA, 64-bits RSA and 128-bits RSA are shown. For different RSA, the time needed to identify timing cases are varies: for the initial few time classes, they are obtained quickly while for the last few time classes they need a very long time.

Our method can correctly identifies all timing classes using formal methods. i.e. for the 32-bits RSA verilog implementation, it identifies all the timing classes with cycle times from 33 to 64. As for the mitigation method mentioned in Figure 4. Since the counter need to count to 64, we only need a 7 bits counter which incurs an approximate area penalty of 7 flops. In contrast, the path-balanced scheduling strategy would require about 512 flip flops. Clearly, with a 64-bit RSA, the savings are more significant. As mentioned earlier, the Compliance State Machine is not in the critical path and incurs no penalty in the operational speed of the circuit.

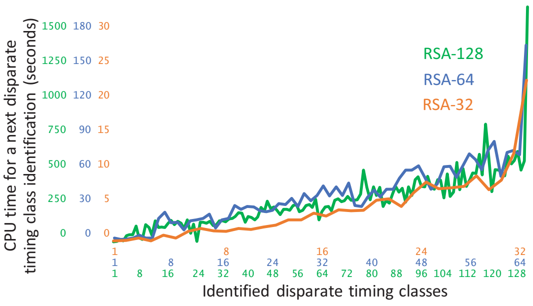


Fig. 5: Normalized Execution Times

VI. Conclusion

Significant numbers of hardware IPs or crypto accelerators are being deployed with the widespread adoption of IoT. It is vitally important that these IPs are *provably secure*. We have proposed a novel approach to discover timing SCA vulnerabilities that (can)exist in designs. This flow also (automatically) eliminates the information leakage caused by the timing channel. The insertion of a lightweight Compensator Block removes the timing channel with minimum modifications to the design with **no impact** on the clock cycle time or combinational delay of the critical path in the circuit. For the future work, multiple secrets in design or multiple public interfaces will be studied. And we will also integrate this framework to High Level Synthesis flow so that more accurate estimates of area can be obtained.

Acknowledgements

This research was supported in part by projects H2020 MSCA ITN RESCUE funded from the EU H2020 programme under the MSC grant agreement No.722325, by the Estonian Ministry of Education and Research institutional research grant no. IUT19-1 and by European Union through the European Structural and Regional Development Funds.

References

[1] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, no. 5, pp. 236–243, 1976.
[2] J. Ming, D. Wu, G. Xiao, J. Wang, and P. Liu, "Taintpipe: Pipelined symbolic taint analysis," in *24th USENIX Security Symposium (USENIX Security 15)*, Washington, D.C., 2015, pp. 65–80.
[3] F. Koeune and F.-X. Standaert, "Foundations of security analysis and design iii," A. Aldini, R. Gorrieri, and F. Martinelli, Eds., 2005, ch. A Tutorial on Physical Security and Side-channel Attacks, pp. 78–108.
[4] B. Coppens, I. Verbauwhede, K. D. Bosschere, and B. D. Sutter, "Practical mitigations for timing-based side-channel attacks on modern x86 processors," in *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*, 2009, pp. 45–60.
[5] K. Tiri and I. Verbauwhede, "A vlsi design flow for secure side-channel attack resistant ics," in *Design, Automation and Test in Europe*, March 2005, pp. 58–63 Vol. 3.
[6] F. Menichelli, R. Menicocci, M. Olivieri, and A. Trifiletti, "High-level side-channel attack modeling and simulation for security-critical systems on chips," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 3, pp. 164–176, July 2008.
[7] A. Ardeshiricham, W. Hu, J. Marxen, and R. Kastner, "Register transfer level information flow tracking for provably secure hardware design," in *Proceedings of the Conference on Design, Automation & Test in Europe*, ser. DATE '17, 2017, pp. 1695–1700.
[8] M. Bidmeshki and Y. Makris, "Toward automatic proof generation for information flow policies in third-party hardware ip," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, vol. 00, 2015, pp. 163–168.
[9] D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers, "A hardware design language for timing-sensitive information-flow security," *SIGPLAN Not.*, vol. 50, no. 4, pp. 503–516, Mar. 2015.
[10] S. Deng, W. Xiong, and J. Szefer, "Cache timing side-channel vulnerability checking with computation tree logic," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '18, 2018, pp. 2:1–2:8.
[11] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the aes s-box," in *Fast Software Encryption*, H. Gilbert and H. Handschuh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 413–423.
[12] L. S. Indrusiak, J. Harbin, and M. J. Sepúlveda, "Side-channel attack resilience through route randomisation in secure real-time networks-on-chip," *CoRR*, vol. abs/1607.03450, 2016.
[13] Z. Jiang, S. Dai, G. E. Suh, and Z. Zhang, "High-level synthesis with timing-sensitive information flow enforcement," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '18, 2018, pp. 88:1–88:8.
[14] JasperGold Security Path Verification App. [Online]. Available: https://www.cadence.com/content/cadence-www/global/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform.html
[15] S. Peter and T. Givargis, "Towards a timing attack aware high-level synthesis of integrated circuits," in *34th IEEE International Conference on Computer Design, ICCD 2016, Scottsdale, AZ, USA, October 2-5, 2016*, 2016, pp. 452–455.

Appendix 4

IV

X. Lai, M. Jenihhin, G. Selimis, S. Goossens, R. Maes, and K. Paul, “Early rtl analysis for sca vulnerability in fuzzy extractors of memory-based puf enabled devices,” in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pp. 16–21, IEEE, 2020

Early RTL Analysis for SCA Vulnerability in Fuzzy Extractors of Memory-Based PUF Enabled Devices

Xinhui Lai[1], Maksim Jenihhin[1], Georgios Selimis[2], Sven Goossens[2], Roel Maes[2], Kolin Paul[3]
[1] Department of Computer Systems, Tallinn University of Technology, Estonia
[2] Intrinsic ID, The Netherlands
[3] Department of Computer Science & Engg, Indian Institute of Technology Delhi, India
Email: xinhui.lai@taltech.ee

***Abstract*—Physical Unclonable Functions (PUFs) are gaining attention in the cryptography community because of the ability to efficiently harness the intrinsic variability in the manufacturing process. However, this means that they are noisy devices and require error correction mechanisms, e.g., by employing Fuzzy Extractors (FEs). Recent works demonstrated that applying FEs for error correction may enable new opportunities to break the PUFs if no countermeasures are taken. In this paper, we address an attack model on FEs hardware implementations and provide a solution for early identification of the timing Side-Channel Attack (SCA) vulnerabilities which can be exploited by physical fault injection. The significance of this work stems from the fact that FEs are an essential building block in the implementations of PUF-enabled devices. The information leaked through the timing side-channel during the error correction process can reveal the FE input data and thereby can endanger revealing secrets. Therefore, it is very important to identify the potential leakages early in the process during RTL design. Experimental results based on RTL analysis of several Bose–Chaudhuri–Hocquenghem (BCH) and Reed-Solomon decoders for PUF-enabled devices with FEs demonstrate the feasibility of the proposed methodology.**

Keywords - timing side-channel attack, physical unclonable function, fuzzy extractor, fault-injection attack, error correction code, BCH, Reed-Solomon, RTL analysis.

I. Introduction

Physical unclonable functions (PUFs) are hardware primitives which derive identifiers and cryptographic keys from the random variations of the silicon manufacturing process. PUFs provide a significantly higher security assurance as keys are volatile and derived only when required. Thus, a PUF can be easily attached or embedded into the cryptographic implementation for authentication and identification [1]. PUF-enabled devices are also an efficient alternative to the expensive conventional measures against the integrated circuit power-off, e.g., by using the Non-Volatile Memory (NVM) for the key storage. The keys generated by PUFs are derived by measurements in the field during the run time and can be saved in a cheaper volatile memory.

PUFs are known to be sensitive to the environmental factors such as the ambient temperature, the supply voltage noise, etc. that may affect the reliability of the response measurement, and ultimately, reduce the reproducibility of the cryptographic key. Along with the external factors, the internal factors of the PUF's manufacturing technology prevent it from guaranteeing a constant response all the time. This nondeterminism poses issues for applying a PUF as a key generator or identifier [2]. Therefore, for the post-processing, a Fuzzy Extractor (FE) is an essential component to help a PUF generate a reliable key by correcting the errors caused internally or by environmental variations.

Different types of the PUF structure and the environmental conditions imply different requirements for the FE and the corresponding ECC. An example of a silicon PUF is the memory-based PUF, which is widely used in chip-level authentication. FE ECCs such as the Bose–Chaudhuri–Hocquenghem (BCH) [2] or Reed-Solomon [3] are used in memory-based PUF enabled devices.

While FEs with ECCs significantly raise reliability, they can lead to new exploits such as allowing an attacker to extract sensitive information by studying the behavior of ECC. Side-Channel Attacks (SCA) on ECC implementations have attracted particular attention of the research community. In [4], the authors extract the information about the key by non-invasive measurement of electromagnetic radiation together with a differential power analysis of the BCH decoder. In [5], the authors study the simple power analysis of both BCH and Reed-Solomon code and manage to recover the PUF response from the collected power traces. However, there is no research work that refers to attacks that combine timing SCA and fault attacks for FEs, namely targeting to the execution time of the error-correcting code of FE in combination with the insertion of faults to PUF. So in this paper, we address this gap by a study on BCH and Reed-Solomon RTL designs execution time differences as a reaction to intentionally triggered faults inserted to PUF. Specifically, the contributions of the paper include:

- Definition of an attack model based on fault injection and timing analysis of ECC execution that may lead to the secret PUF values extraction.
- An early design stage RTL methodology for verification of an ECC design invulnerability against the proposed attack by employing both structural and simulation-based analysis steps.
- Case studies of Reed-Solomon and BCH based ECC with vulnerabilities identification and exploitation.

The rest of the paper is organized as follows. Section II reviews the background of the FE architecture and ECC decoders. The attack model is discussed in Section III. Section IV presents the proposed methodology for verifying invulnerability against the proposed attack. Section V presents a case study for ECC implementations. Section VI concludes the paper.

II. Background And Related Works

A. Fuzzy Extractor and Secure Sketch

The Fuzzy Extractor [6] is a secure method to generate cryptographic keys from noisy sources. The FE serves as a post-processing unit in memory-based PUF-enabled cryptographic schemes. It is used both in the Generation and Reconstruction Procedures, as illustrated in Fig. 1 and Fig. 2 correspondingly.

In the Generation Procedure case, the fuzzy data from the PUF response W and a random secret S are used to generate the Helper Data by XOR operation on W and $E(S_0)$ which is encoded S_0. The generated helper data is stored in a non-volatile memory. In memory-based PUF-enabled devices, the Generation Procedure happens only once at the first-time power-on of the memory-based PUF.

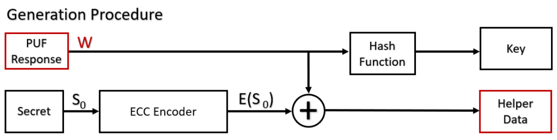


Fig. 1. Generation Procedure in A PUF Fuzzy Extractor

On the contrary to this, the Reconstruction Procedure is executed many times during the product lifetime. Due to the noise and PUF manufacturing randomness, it is difficult to generate the same response consistently. To reproduce the correct cryptographic key, the Helper Data, stored in an NVM, is used in conjunction with the measured PUF response W'. Then with the help of the ECC decoder to detect and correct the divergent bits, the correct W is reproduced. After applying the Hash Function, the expected correct cryptographic key is reconstructed.

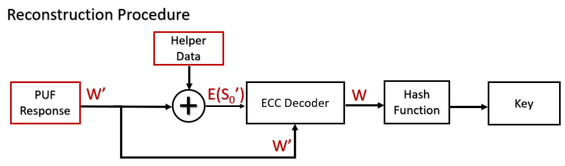


Fig. 2. Reconstruction Procedure in A PUF Fuzzy Extractor

The FE guarantees that the resulting key is consistent while the publicly accessible Helper Data does not leak any information related to the secret of the key. To ensure consistent generation of the correct key, the hamming distance between the measured PUF response W' with the originally measured W in the Generator Procedure should be smaller or equal to the correction capability of the ECC decoder, represented as a constant value $\boldsymbol{t}$. In this paper, we assume that the measured responses of the memory-based PUF are within this hamming distance constraint.

Recent research works have identified potential attacks on FEs [7]. Most of them target the Reconstruction Procedure. In [8], the authors report on a method to extract the PUF secret by manipulating the Helper Data in the Reconstruction Procedure. In [9], Delvaux et al. provide an in-depth analysis of the Helper Data algorithms, and identify new threats for leaking the Helper Data and the soft-decision coding.

B. ECC decoder

The ECC unit is the main component in a FE. Binary BCH and Reed-Solomon are the two types of ECC that are widely used in PUF-enabled devices. Both codes are cyclic and capable of detecting up to $2t$ and correct up to t errors by adding $2t$ check bits or non-binary values (symbols) to the data. Binary BCH is used for binary error correction, and Reed-Solomon is used for symbol error correction. While both software and hardware implementations of these codes exist, the hardware ones are more adopted. First, this is because the complex algorithms of the decoders require significant computational power along with the real-time constraints. The second difficulty for software implementations is the limited support of the Galois Fields Arithmetic operation in the general-purpose processors [10]. The hardware implementations of binary BCH and Reed-Solomon decoders are discussed in more detail in Section V.

III. Attack Model

In this paper, we assume an attack combining 1) fault injection to the memory-based PUF with 2) a timing SCA for observing and comparing the different decoding execution times of the ECC unit that is aimed at revealing the correct memory-based PUF data. In case of success, the attack explicitly compromises the core function of the PUF-enabled cryptographic devices, because the attacker can clone the PUF and can steal the secret.

A. Fault Injection Parameters

For the physical fault injection to the memory-based PUF the following fault parameters are assumed.

(a) Granularity: each fault injection results in exactly one fault in one-bit data.
(b) Modification (fault type): after the fault injection, the manipulated data is set to a specified logic value, i.e. either '1' or '0'.
(c) Control: the attacker has a bit-wise precise control of fault injection to the memory-based PUF bits.
(d) Effect of the fault: the injected faults have a transient nature, i.e. the injected values are overwritten by the normal functionality of the device (e.g. the next measurement of the PUF on power-on).

Several studies on laser fault injection [11] have demonstrated similar attack parameters and, therefore, the feasibility of the above assumptions. Technical details of the fault injection attack implementation are out of the scope of this study.

B. *Attack Assumptions*

The following set of assumptions must be satisfied for the success of the attack. The feasibility of the assumptions (iii)-(vi) is supported by several research works in state of the art.

(i) The output of a memory-based PUF measurement in the cryptographic device is processed by a FE with a binary BCH or Reed-Solomon based ECC.

(ii) The ECC implementation leaks exploitable information through the timing-side channel.
Comment: The methodology for identifying the vulnerability enabling this assumption is the core contribution of this paper and presented in Section IV.

(iii) The memory-based PUF is noise-free under stable environmental conditions. The errors in the memory-based PUF are caused by the environment.
Comment: While an ideal noise-free memory-based PUF would not require the FE at all, we assume that the noise is caused by the variations in the external environment while the internal noise is negligible. [12] demonstrated that the external environmental conditions like the ambient temperature, supply voltage, etc. have a significant impact on the error rate of the PUF.

(iv) The generated Helper Data is stored in NVM or the flash memory of the cryptographic devices and remains constant during the Reconstruction Procedure.
Comment: As an added value, this assumption creates an advantage for the proposed attack, compared to alternatives (e.g. [8], [9]), because it does not rely on the attacker being able to modify the Helper Data.

(v) The fault injection parameters (a) to (d) hold (see III.A).
Comment: Several research works proposed bit-wise fault injection in SRAM and other on-chip memories. E.g., in [13], bit-wise faults were successfully injected in a PIC microcontroller through a semi-invasive method and without mechanical damage to the silicon.

(vi) The attacker has a controlled access for measuring the decoding execution time.
Comment: The physical measurement of the ECC decoding execution time can exploit the reflection of timing by the power traces. In [14], the authors analyze use of the AES execution power traces for a SCA. The power traces are represented by changes of power over time, with the timing information embedded. A similar approach is used in [15] for RTL verification of RSA designs against vulnerability to timing SCAs.

C. *Attack Procedure*

The proposed attack is a combination of fault injection with timing side channel analysis and represented by the following 4 steps. The procedure is illustrated in Fig.3.

1) Power on the device. Measure the initial PUF data. With the above assumptions, this memory value should be error-free, i.e. the same with W generated in the Generation Procedure. Measure and record the reference time T as the number of clock cycles for the execution of the ECC decoding.
2) Inject a fault f at the m_{th} bit of memory-based PUF following the (a) to (d) parameters and generate the new memory data W_{m_f}. W_{m_f} has a one-bit difference value compared to W. E.g, if the f is a set to logic "1" value and $m = 1$ then W and W_{1_f} can be either equal or can be different by exactly one bit at the first position. Then execute the Reconstruction Procedure, measure the decoding execution time $T(m)$.
3) The relation between these two decoding times T and $T(m)$ contains only two possible cases. The PUF's secret single bit m can be revealed by comparing the two decoding times as follows:
   - if $T! = T(m)$, then a different value at the m_{th} bit was injected. E.g., for $f = 1$, the original value of the m_{th} bit in memory is '0';
   - if $T = T(m)$ then the value at the m_{th} bit was equal to the injected one. E.g., for $f = 1$, the original value of the m_{th} bit in memory is '1';
4) Repeat the steps 1) to 3). of the procedure until the last m_{th} bit of memory-based PUF. The memory-based PUF's secret value is revealed.

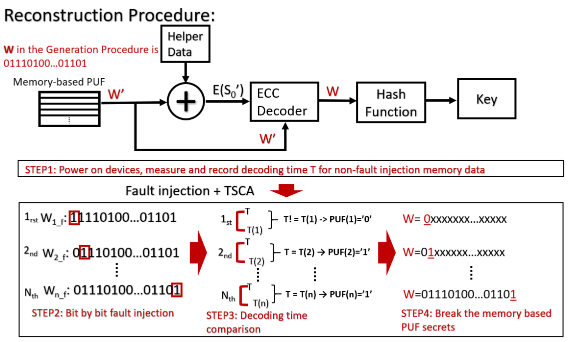

Reconstruction Procedure:
W in the Generation Procedure is 01110100...01101
Helper Data
Memory-based PUF
W'
E(S0')
ECC Decoder
W
Hash Function
Key
W'
STEP1: Power on devices, measure and record decoding time T for non-fault injection memory data
Fault injection + TSCA
1rst W1_f: 11110100...01101
2nd W2_f: 01110100...01101
Nth Wn_f: 01110100...01101
STEP2: Bit by bit fault injection
1st T T(1) T! = T(1) -> PUF(1)='0'
2nd T T(2) T = T(2) → PUF(2)='1'
Nth T T(n) T = T(n) → PUF(n)='1'
STEP3: Decoding time comparison
W= 0xxxxxxx...xxxxx
W=01xxxxxx...xxxxx
W=01110100...01101
STEP4: Break the memory based PUF secrets

Fig. 3. An illustration of the proposed attack procedure

IV. Proposed Methodology

The precondition for the introduced attack is the non-constant decoding execution time in case of different input data for the ECC unit of the memory-based PUF Fuzzy Extractor. In this section, we propose a methodology to identify this vulnerability in an ECC implementation already at the RTL design phase. The methodology employs both structural and simulation-based analysis for binary BCH and Reed-Solomon algorithms based hardware ECC implementations. In practice, these two algorithms are widely used by the industry in memory-based PUF-enabled devices.

A. *Structural Analysis of ECC Decoder*

1) Binary BCH Decoder: A general binary BCH decoder hardware implementation has three stages, as shown in Fig.4. The divergent (error) bits are identified by the Syndrome Calculator, Key Equation Solver and the Chien Search. Next, the decoder corrects the error bits by the XOR operation on the stored input with the identified error bits to recover the correct codeword. Let r(x), c(x) and e(x) be the received polynomial, codeword polynomial and error polynomial, i.e. r(x) = c(x) + e(x). Assume the binary BCH decoder can correct t errors. As the structural analysis of the binary BCH, we consider the following reasoning.

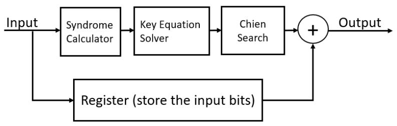


Fig. 4. Binary BCH Decoder Structure

- **Syndrome Calculator:** It is the first stage in the decoder generates 2t syndromes as defined in (1).
  $$S_i = r(x^i) = r_0 + r_1 x^i + r_2 x^{2i} + ... + r_{n-1} x^{(n-1)i} \quad (1)$$
  where $1 \leq i \leq 2t - 1$.An important feature of the syndromes is that they do not depend on transmitted information but only on error locations. If at position i there is an error then S_i has a non-zero value and it is equal to zero otherwise. For all possible inputs, the decoder always generates $2t$ syndromes. Therefore, the time for the syndrome calculation is constant for the BCH decoder with a fixed error correction capability.
- **Key Equation Solver:** In the second stage, the error location polynomial $\sigma(x)$ is generated. Berlekamp Massey Algorithm (BMA) is one known iterative procedure that determines polynomial equation (2) out of a set of linear equations for the 2t syndromes calculated in the first stage.
  $$\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + ...\sigma_t x^t \quad (2)$$
  BMA can be implemented in parallel or serially. In [16], it is demonstrated that a parallel implementation for a t errors correction BMA needs 2t iterations. A serial implementation implies a significant increase in the number of iterations. According to [17], it needs $2t^2$ iterations. However, for both cases, the total number of iterations is determined only by t, which is the maximum number of errors the decoder can correct.
- **Chien Search:** This stage searches for error locations by checking the roots of $\sigma(x)$. It is a simple trial-and-error procedure. All nonzero elements of the Galois Fields for a binary BCH decoder are generated in sequence and only capture the condition when $\sigma(x_i)$ is equal to zero which the error position. Therefore, in this stage, the total number of nonzero elements depends only on the Galois Field GF(2^m) where $n = 2^m - 1$ and n is the size of codeword.

To conclude, for different binary BCH decoder implementations, the error correction bits and the size of the codeword are the factors which lead to the different decoding execution time. However, for a specific binary BCH decoder, these parameters are fixed at the design phase. Therefore, the structural analysis has not identified timing channels in binary BCH decoder structures.

2) Reed-Solomon Decoder: Reed-Solomon (RS) decoder aimes at non-binary (symbol) error correction. Different from the binary BCH, which needs only to generate error locator polynomial $\sigma(x)$ RS also needs to generate an error value polynomial. Therefore, some RS implementations replace BMA by Euclidean Algorithm (EA) for the Key Equation Solver to calculate the error location polynomial and error value polynomial and add a new component Forney to calculate the error value. The Reed-Solomon decoder structure is illustrated in Fig.5. Here, the differences with the BCH decoder structure are highlighted in red. In the following structural analysis, we focus only on these two different components.

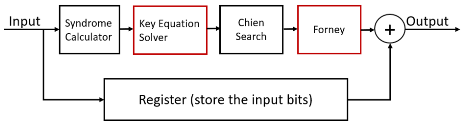


Fig. 5. Reed-Solomon Decoder Structure

- **Euclidean Algorithm (EA):** It is an iterative procedure to generate the *error locator polynomial* and the *error value polynomial* with the 2t syndromes generated by the Syndrome Calculator stage. Particular implementations of EA may prefer a pipelined version with the objective of performance optimization [18]. In EA procedure [18], the error locator polynomial $\sigma(x)$ and the error value polynomial $\omega(x)$ are acquired by solving the equation (3). Equation (3) can be represented in the form of equation (4). The extend Euclidean Algorithm can find a series polynomial by (5). From (4) and (5), $A_i(x) = \sigma(x)$, $R_i(x) = \omega(x)$ and $B_i(x) = -Q(x)$. To solve the Key Equation the EA procedure starts with initiating the values $R_0(x) = x^{2t}$, $Q_0(x) = S(x)$, $L_0(x) = 0$, $U_0(x) = 1$ and then it is followed by interactions of four equations used to calculate $R_i(x)$, $Q_i(x)$, $L_i(x)$ and $U_i(x)$, based on the values from the previous stage, until the degree of $R_i(x)$ gets smaller than the degree of $L_i(x)$ or t. When the iteration is finished, the equation (3) is solved. Because the $R(x)$ starts at the degree 2t, and the iteration can finish at the degree of $R(x)$ equal to t or smaller. Therefore, the EA stage may require a different number of iterations for the different codewords which may introduce different execution times.
  $$\omega(x) = S(x)\sigma(x) \mod x^{2t} \quad (3)$$
  $$\sigma(x)S(x) = Q(x)x^{2t} + \omega(x) \quad (4)$$
  $$A_i(x)S(x) + B_i(x)x^t = R_i(x) \quad (5)$$
- **Forney:** By using the Forney algorithm, the error value e(x) can be acquired by the equation (6).
  $$e_j = -\frac{\omega(X_j)}{\sigma'(X_j)} \quad (6)$$
  Normally, it is implemented in combinational logic because $\omega(X)$ and $\sigma(x)$ are available. The execution time of this stage is constant.

To conclude, the structural analysis has not identified the timing channel in the other stages of the Reed-Solomon structure but the second stage. Based on the implementation, the Key Equation Solver stage in the Reed-Solomon based ECC decoder can introduce the vulnerability.

B. Simulation-based analysis of ECC decoder

In an RTL simulation of an ECC decoder implementation, a number of stimuli data parameters may have an impact on the execution time of a decoding iteration. For the proposed simulation-based analysis step, the following parameters are identified:

- $codeword_{value}$: the encoded codeword value
- $error_{value}$: the error value is relevant only for a non-binary (symbol) ECC decoders
- $error_{position}$: the error bit position for a binary ECC decoder or the error symbol position for a non-binary ECC decoder
- $error_{number}$: the number of error bits or symbols for binary or non-binary ECC decoder correspondingly

The structural analysis of binary BCH and RS decoders and the defined attack model allows reducing the search space. Table I presents the relationship of the execution time variation introduced by manipulating a particular decoding parameter and the vulnerability to the proposed attack. The notations *C* and *NC* represent constant and non-constant decoding execution time, while *V* and *NV* represent vulnerability or invulnerability.

TABLE I
ECC EXECUTION TIME VARIABILITY AND THE SCA VULNERABILITY

ECC Decoding Execution Time/Vulnerability		
Parameters	RS decoder	Binary BCH decoder
$codeword_{value}$	C/NV	C/NV
$error_{value}$	C/V	
$error_{position}$	C/V	C/V
$error_{number}$	NC/V	C/V

In particular, manipulation of the $codeword_{value}$parameter does not identify the vulnerability of the target decoder. The attacker does not have access to manipulate the predefined correct codeword and can only manipulate the input codeword to cause an error. Based on the structural analysis, it is already known that different codewords do not introduce different decoding time neither in binary BCH nor in RS structures. The $error_{value}$ and $error_{position}$ parameters can be manipulated by the attacker by injecting faults to the input codeword. However, the constant decoding time will not leak information through the timing channel. From Table I, we can conclude that the binary BCH decoder structures are secure with regards to the information leakage through the timing channel. An RS decoder implementation can be vulnerable if the attacker injects a different number of error symbols, i.e. the $error_{number}$. The table guides the designer which simulation campaigns are required to verify a particular implementation against vulnerability to the proposed SCA.

V. CASE STUDY

The feasibility of the proposed methodology was validated by running an exhaustive simulation campaign on 3 case study ECC designs for memory-based PUF Fuzzy Extractors, i.e. 2 binary BCH and a Reed-Solomon ECC implementations.

A. Binary BCH decoder

The implementation of the binary BCH decoder is an open-source design in RTL Verilog accessible from Github [19]. Its general architecture is illustrated in Fig. 4. The decoder was configured for a 12-bit codeword, 8-bit message and supports two types of BMA, i.e. serial *BMA_serial* and parallel *BMA_parallel* versions. The configuration was set to correct up to two errors, i.e. $t = 2$. Both versions were simulated with an exhaustive set of test vectors to identify the timing information leakage. Only valid values for the 12-bit binary codeword were extracted by running the encoder with all possible inputs. The input for the encoder is 4-bit message and 2-bit error correction capability. Since the number of errors correctable for a given polynomial is sparse, the encoder has the selection algorithm to select suitable polynomial function to meet the provided requirements. Thus the actual message bit might be changed. In our case, the encoder pads 4-bit zeros and makes the input message bit 8-bit. We input all possible 4-bit value into encoder. Then each encoded message value was merged with all possible error combinations considering the injection of 0, 1 or 2 errors at a time, i.e. all combinations of $error_{number}$ and $error_{position}$ were simulated. This means $T_{test\_vectors} = 2^4 * (\binom{12}{0} + \binom{12}{1} + \binom{12}{2})$=1,264 ECC decoding executions were analyzed for the each design, and the decoding time was measured.

B. Reed-Solomon decoder

The case-study Reed-Solomon decoder implementation is also an open-source design accessible from Github [20] and illustrated in Fig.5. The design was configured for 8-symbol codewords, 4-symbol messages and 8-bit symbols. The error correction capacity was also set to 2 errors, i.e. $t = 2$. By default, the design is pipelined by using registers to extend the execution time for each stage to the worst execution-time case. In practice, for memory-based PUF enabled devices where execution time is a critical factor, a configuration aimed at the decoder speed optimization is often used. This was also applied for the current case study. Different from the binary BCH, the Reed-Solomon decoder uses symbol-based error correction. While the parameter $error_{position}$ represents the position of the error symbol, the $error_{value}$ can take one of the $2^8 = 256$ possible values for an error in each symbol. The number of all combinations for the valid codewords merged with all possible errors for each symbol is $T_{test_vector} = \binom{8}{1} * (2^8 - 1) + \binom{8}{2} * (2^8 - 1) + \binom{8}{0}$= 1,822,741 that represents the number of executions to simulate and analyse per codeword. In the simulation campaign, we limited the analysis to one random valid codeword. Based on the architecture analysis, the other codewords provide the same results.

C. Experiment Results Analysis

Experiment results are shown in Table II. In the list of parameters identified for manipulation by the proposed methodology, the symbols "●" and "-" represent the varied and constant parameters correspondingly. T_d denotes the number of different decoding execution times identified and the corresponding values in clock cycles. For the Binary BCH, the experimental results confirm the conclusions of the structural analysis and do not identify any variations in the execution times. For the Reed-Solomon decoder, the red cells highlight the cases with the varying decoding time. In this experiment, T_d*:3* $\{$*38, 66, 72*$\}$ denotes different timing cases in case of the different number of errors to be corrected, i.e. 38, 66 or 72 clock cycles for 0, 1 or 2 errors correspondingly. As shown in the first three rows, different $error_{position}$ and $error_{value}$ can not affect the decoding time, and it remains constant (but can be equal to different values) $T_d : 1\ \{38\}\|\{66\}\|\{72\}$.

TABLE II
ECC-BASED FE DECODING TIMING ANALYSIS

Varied Parameters				Decoding time by ECC Implementations (clock cycles)		
$codeword_{value}$	$error_{number}$	$error_{position}$	$error_{value}$	Binary BCH-12-8 BMA_serial	Binary BCH-12-8 BMA_parallel	Reed-Solomon-4-8-8
-	-	-	●			T_d:1 $\{38\}\|\{66\}\|\{72\}$
-	-	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	T_d:1 $\{38\}\|\{66\}\|\{72\}$
-	-	●	●			T_d:1 $\{38\}\|\{66\}\|\{72\}$
-	●	-	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	T_d:3 $\{38, 66, 72\}$
-	●	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	T_d:3 $\{38, 66, 72\}$
-	●	-	●			T_d:3 $\{38, 66, 72\}$
-	●	●	●			T_d:3 $\{38, 66, 72\}$
●	-	-	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	
●	●	-	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	
●	●	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	
●	-	●	-	T_d:1 $\{28\}$	T_d:1 $\{21\}$	

VI. CONCLUSIONS

Application of Fuzzy Extractors for error correction may enable opportunities to break the secure PUFs if no countermeasures are taken. This paper considers a combined attack model based on fault injection and timing analysis of ECC execution. In the worst case, such an attack may lead to the secret PUF value extraction. An early design stage RTL methodology was developed to verify the ECC design invulnerability against such or a similar SCA.

The methodology involves structural and simulation-based analysis parts. In our study, we targeted at two ECC architectures most widely used in FEs. The structural analysis has not identified vulnerabilities in the considered binary BCH architectures, while the architecture of Reed-Solomon based ECC may be vulnerable in particular implementations. A set of simulation-based experimental results have confirmed the findings and demonstrated the timing information leakage. Under the specified assumptions, the proposed attack procedure is able to exploit this vulnerability and reveal the secret.

The results of the early RTL analysis can guide in the selection of suitable ECC implementation or in the application of design-level countermeasures. To remove the leakage, e.g., a register can be added at the output of the Euclidean Algorithm stage to equalize the timing to the worst-case execution, or optimizations at the ECC algorithm may be applied. The efficiency of the mitigation solutions can be explored by the proposed methodology at a low cost.

VII. ACKNOWLEDGEMENTS

This research was supported in part by the project H2020 MSCA ITN RESCUE funded from the EU H2020 programme under the MSC grant agreement No.722325 and by European Union through the European Structural, Regional Development and Social Funds.

REFERENCES

[1] R. Maes *et al.*, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*. Springer, 2010, pp. 3–37.

[2] R. Maes *et al.*, "A soft decision helper data algorithm for sram pufs," in *2009 IEEE international symposium on information theory*.

[3] A. R. Korenda *et al.*, "A proof of concept sram-based physically unclonable function (puf) key generation mechanism for iot devices," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2019, pp. 1–8.

[4] L. Tebelmann *et al.*, "Em side-channel analysis of bch-based error correction for puf-based key generation," in *Proceedings of the 2017 Workshop on Attacks and Solutions in Hardware Security*.

[5] D. Karakoyunlu *et al.*, "Differential template attacks on puf enabled cryptographic devices," in *2010 IEEE International Workshop on Information Forensics and Security*. IEEE, 2010, pp. 1–6.

[6] Y. Dodis *et al.*, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004.

[7] D. Merli *et al.*, "Side-channel analysis of pufs and fuzzy extractors," in *International Conference on Trust and Trustworthy Computing*. Springer, 2011, pp. 33–47.

[8] G. T. Becker. (2017) Robust fuzzy extractors and helper data manipulation attacks revisited: Theory vs practice.

[9] J. Delvaux *et al.*, "Helper data algorithms for puf-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2014.

[10] M. Riley *et al.*, "An introduction to reed-solomon codes: principles, architecture and implementation," 2003.

[11] C. Roscian *et al.*, "Fault model analysis of laser-induced faults in sram memory cells," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2013, pp. 89–98.

[12] Y. Gao *et al.*, "Building secure sram puf key generators on resource constrained devices," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*.

[13] S. P. Skorobogatov *et al.*, "Optical fault induction attacks," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 2–12.

[14] A. Krieg *et al.*, "A side channel attack countermeasure using system-on-chip power profile scrambling," in *2011 IEEE 17th International On-Line Testing Symposium*. IEEE, 2011, pp. 222–227.

[15] X. Lai *et al.*, "Pascal: Timing sca resistant design and verification flow," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2019, pp. 239–242.

[16] W. Liu *et al.*, "Low-power high-throughput bch error correction vlsi design for multi-level cell nand flash memories," in *2006 IEEE Workshop on Signal Processing Systems Design and Implementation*.

[17] H.-C. Chang *et al.*, "New serial architecture for the berlekamp-massey algorithm," *IEEE transactions on communications*, 1999.

[18] S. Lee *et al.*, "A high-speed pipelined degree-computationless modified euclidean algorithm architecture for reed-solomon decoders," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 91, no. 3, pp. 830–835, 2008.

[19] "Verilog based bch encoder / decoder," https://github.com/russdill/bch_verilog.

[20] "Freecores reed-solomon codec generator," https://github.com/freecores/reed_solomon_codec_generator.

Appendix 5

V

X. Lai, T. Lange, A. Balakrishnan, D. Alexandrescu, and M. Jenihhin, "On antagonism between side-channel security and soft-error reliability in bnn inference engines," in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2021

On Antagonism Between Side-Channel Security and Soft-Error Reliability in BNN Inference Engines

Xinhui Lai[1], Thomas Lange[2,3], Aneesh Balakrishnan[1,2], Dan Alexandrescu[2], Maksim Jenihhin[1]
[1] Department of Computer Systems, Tallinn University of Technology, Estonia
[2] IROC Technologies, France
[3] Dipartimento di Informatica e Automatica, Politecnico di Torino, Italy
Email: xinhui.lai@taltech.ee

Abstract—**Recently, several research works have emphasized the problem of stealing the intellectual property of trained Machine Learning (ML) models from hardware neural network inference engines spotlighting Binarized Neural Networks (BNNs). The binary operations in BNNs can be executed bitwise, which notably saves storage memory, reduces the execution time and power and, therefore, makes them convenient for implementation in hardware. Unfortunately, these advantages may also enable a vulnerability to Differential Power Analysis (DPA) side-channel attacks, which, in turn, necessitates dedicated masking techniques to protect the models. Notably, the recent BNN hardware inference engines are being increasingly adopted for critical applications and demand, along with security, also high levels of in-filed reliability throughout their lifetime. The state-of-the-art power side-channel masking in BNNs implies glitch-resistant structures, such as Trichina AND gates and sequences of flip-flops, and may create soft-error reliability issues that are currently overlooked in the literature. This paper presents an analysis for the soft-error reliability jeopardy by the security countermeasures in hardware implementations of BNN inference engines. Our work reveals a steep increase (hundreds of times) of vulnerability to single-event effects, introduced by the state-of-the-art security enhancement techniques, and emphasizes the interdependency of the design's reliability and security aspects.**

Keywords - Binarized Neural Network (BNN), soft-error reliability, logical de-rating, side-channel attack, Differential Power Analysis (DPA).

I. Introduction

Deep Neural Networks (DNNs) are designed to classify objects after training and have been proven effective in many Artificial Intelligence (AI) applications. To improve the energy efficiency and throughput, a trained DNN model can be mapped to a hardware inference engine [1]. As a rule, generating an industrial DNN model implies a significant amount of training data, computing resources as well as human efforts. Therefore, it is essential to protect a DNN model against data and functionality integrity violations, intellectual property rights and potential illegitimate reproduction [2]. On the other hand, the DNN inference engines are increasingly adopted for critical applications demanding high levels of functional safety and reliability to in-field faults.

A Binarized Neural Network (BNN) is a particular class of DNNs proposed in [3] characterized by binary weights and activation functions that require less storage and computational resources compared to full-precision DNN models. Therefore, BNN hardware inference engines are efficient for applications in critical embedded systems [4], e.g. autonomous robotic vehicles and ML-powered edge devices. However, recent research works have discovered that the simpler binary operations in BNN have accidentally created power side-channels. In a recent research work [5], Anuj Dubey, et al. show that the Differential Power Analysis (DPA) [6] on a running BNN hardware inference engine can extract the secret model parameters such as weights. Using the methodology described in [7], the authors repeatedly apply the DPA method on a 4-bit secret weight of a BNN. Based on 100k measurements, they compute the corresponding power consumption of the intermediate computation for all the 16 possible values and discover a significant correlation between the power measurements and the correct weight values, which can be exploited for information leakages. To address DPA in BNN hardware inference design, data masking techniques are proposed in [8]. In the paper, the authors propose a fully-masked BNN hardware inference engine design by masking all the linear and non-linear operations that existed in the implementation and validate the effectiveness on an FPGA with 2M power traces.

Recently, BNN hardware implementations are increasingly employed for critical applications with the requirement for in-field lifetime reliability and trustworthy results even in the presence of hardware-level faults [9]. Unfortunately, the described above security enhancements modifying the logic structure of the design may negatively affect the soft-error reliability of the DNN. To the best of our knowledge, the effects of such side-channel masking techniques on soft-error reliability of the hardware design are not duly addressed in the literature.

In this work, we propose a soft-error reliability analysis of the security-enhanced BNN. We apply an existing masking technique to the BNN hardware implementation and prove that the reliability of the DPA-resistant HW BNN inference engine is compromised.

The paper makes the following main contributions:

- Establishing a novel discussion for reliability jeopardy introduced by side-channel security countermeasures in the HW BNN inference engines.

- Proposing a soft-error reliability analysis flow for HW BNN inference engines based on logical and functional de-rating factors evaluation.
- Proving a significant soft-error increase vulnerability in a representative case-study open-source BNN design, specifically, up to 1000$\times$ output bitwise failure rate increase or up to 350$\times$ neural network functional failure rate increase.

The rest of the paper is organized as follows. Section II reviews the background of BNNs, the relevant power side-channel attacks and the corresponding mitigation techniques. In section III, the soft-error reliability assessment is discussed. Section IV explains the case-study BNN architecture and the performed reliability analysis, followed by experimental results in section V. Section VI concludes the paper.

II. Background And Related Works

A. *Binarized Neural Network*

Generally, a BNN has an input layer with input vectors, several hidden layers composed of internal operational neurons, and an output layer that computes the final result (see Fig. 1a). In BNNs, the arithmetic computations use binarization, i.e. the most extreme form of network quantization [10]. Here, the weight multiplication can be replaced by a simple bitwise operation XNOR operation. The summation can be simplified by using a population count (or *popcount*) that calculates the number of bits set to ’1’ in a bit vector (see Fig. 1b).

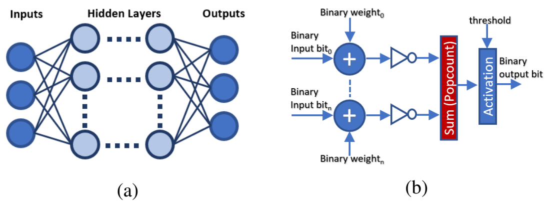


Fig. 1: Binarized Neural Network

B. *Power Side-Channel Attacks on BNNs*

However, the simple structure of such a neural network has its drawbacks. Recent research works have demonstrated BNN model extraction vulnerabilities [11] that exploit physical side channels such as time, power and electromagnetic emanations [12], [13]. Notably, the work in [5] demonstrates an impressive Differential Power Analysis side-channel attack that can reveal the secret weights of a fully connected BNN. Considering that the weight value is binary, the summation is usually implemented by a pipelined adder tree composed of several stages that need additional registers in the middle to store intermediate values. The adversary can target any stage of the adder tree, primarily focusing on the switching activity of the registers in the adder tree to extract the power traces. With the power traces at hand, it may be feasible to steal the weights and biases values of the trained model. In the mentioned example, the authors apply the DPA method [7] on the second stage registers of the adder tree and succeed in extracting the values of the weights and the biases for all nodes by analyzing the Pearson correlation coefficient of the power traces.

C. *BNN Power Side-Channel Masking Techniques*

To address the above vulnerability, [8] proposes techniques for the power side-channel masking aiming at enhancing the HW BNN inference engine’s resistance to DPA. In the paper, the authors analyze the neural network specific computation and propose the following masking techniques:

- For the weighted summations, build a protected AND operation by adopting the glitch-resistant Trichina’s AND gate for the Ripple-Carry Adders (RCA) to replace them with the security-masked N-bit RCA;
- For multiplexers, use Look-Up Table (LUT) as a replacement, e.g., an 1-bit multiplexer is replaced with a 4-input 2-output security-masked LUT.
- For the activation functions, perform the NOT operation by inverting one of the two Boolean shares of MSB received from the previous security-masked adder.
- For the output layer, transform the problem of the masked comparison to masked subtraction and reuse the masked adder as a subtractor.

Among the four mitigation techniques, only the first one, i.e. the RCA masking, introduces numerous extra registers, depending on the size of the adder. This particular technique creates an important case study for our analysis. Intuitively, the introduced massive sequences of flip-flops behave as ”magnets” to additional soft-errors to be caught by the BNN in the field. According to our hypothesis, a BNN protected against security side-channels with this or a similar technique becomes significantly more vulnerable to the soft-error reliability issues. Interestingly, the adopted Trichina AND gates structure [14] is also known in the literature for its application to AES co-processors and similar cryptocores to protect them from power side-channel attacks. This fact potentially extends the impact of our analysis.

III. Soft-Error Reliability Assessment for HW BNN Implementations

A. *De-Rating Based Soft-Error Reliability Assessment*

With the technology feature size shrinking, the probability of electronic systems to experience Single Event Effects (SEEs) is increasing and the overall vulnerability to the radiation-induced transient faults, i.e. soft-errors, is becoming more prominent [15]. To evaluate the reliability to soft-errors for a circuit, the analysis of sensitivity of the underlying cell’s technology is used. However, not all faults occurring in the cells lead to failures, i.e. observable effects at the application level, but can be masked on the way. De-rating factors are used to quantify the masking effects of soft-errors. At logic level, the de-rating analysis relies on logic-level models for SEE, i.e. a Single Event Upset (SEU) for sequential cells and a Single Event Transient (SET) for the gates. Moreover, in an RTL analysis, the SET faults, can be modeled with a reasonable accuracy by a subset of SEUs. The overall Soft-Error Reliability (SER) for SEUs in a system circuit can be expressed by Equation (2), where $\text{SER}_{\text{SEU},i}$ is the rate of SEUs in a sequential cell i (a flip-flop). $\text{SER}_{\text{SEU},i}$ is calculated by Equation (1) [16], [17], where the Failure-In-Time $\text{FIT}_{\text{SEU},i}$ denotes the rate of soft errors for the sequential element and depends on the underlying technology and the operating environment. The considered de-rating factors are Temporal De-Rating (TDR_i), Logical De-Rating (LDR_i), Functional De-Rating (FDR_i), respectively.

$$\text{SER}_{\text{SEU},i} = \text{FIT}_{\text{SEU},i} \cdot \text{TDR}_i \cdot \text{LDR}_i \cdot \text{FDR}_i \tag{1}$$

$$\text{SER}_{\text{SEU}} = \sum_{i \in \text{flip-flops}} \text{SER}_{\text{SEU},i} \tag{2}$$

B. Reliability Assessment for the HW BNN Implementations

Recent research works [9], [18]–[22] have demonstrated the significance of the soft-error reliability study for machine learning systems in safety-critical applications like autonomous unmanned robotic vehicles and AI-powered edge devices. One of the interesting questions in the DNN reliability research is related to understanding the resilience of particular layers to transient and permanent faults. The conclusion in [23] considering faults in registers of a RTL model was that permanent faults in inner layers cause less inference errors than the permanent faults in the first layers. This is in line with the studies of permanent faults in [15] and [24]. However, according to [23], the later layers are more vulnerable to transient faults compared to the first layers. Also the research in [25] and [26] demonstrates that the random bit flips in weights for the early layers has less impact on the inference accuracy compared to the faults injected to the last layers. In our current analysis, as discussed in next section, we follow the arguments that the *last layers* of a DNN are the most critical for the SER assessment.

The studies for DNN reliability assessment include frameworks, e.g Fidelity [21], for abstracting the level of or optimizing the fault analysis for DNN hardware inference engines such as fixed-/floating point CNNs generated by NVDLA, Eyeriss, MAERI, FINN, CAFFEINE, ISAAC and similar generators. Frameworks for automated generation of binarized NN HW inference accelerators from established DNN evaluation and training platforms such as Tensorflow, PyTorch or Caffe are limited. Therefore, the SER analysis flow for the BNN designs (initial ones and the ones containing the security enhancements) employed in our work relies on a two-step approach. First, we perform de-rating evaluation for the critical parts of the case-study BNN, e.g. the security-masked ripple carry adder. Second, we execute a fault injection simulation campaign.

For the first step we rely on the SoCFIT tool by IROC. It is a reliability-focused design characterization tool that predicts various de-rating factors and calculates the failure rate of digital circuits. The tool computes the SEU logic de-rating by analyzing the structure of the RTL description of the circuit. Therefore, all paths from the sequential cell to any end point (primary output) are considered and the LDR for each found paths is evaluated, by multiplying the intrinsic LDR for each gate along the path. The overall LDR for the sequential cell can then be computed by either selecting the value of the path with the maximum path or averaging the results of the individual paths.

C. BNN-level Fault-Injection Simulation Campaign

In this paper, the reliability analysis for HW BNN inference engine implementations is performed by statistical fault injection simulations on the RT level of the design. The BNN circuit is simulated with a logic-level simulator by running a corresponding inference testbench. With the testbench the correct behaviour of the BNN circuit can be verified. This can be achieved by monitoring and recording all outputs of the neural network or testing the functional behaviour of the inference engine. First, a simulation is run without any faults injected to create a reference golden run. Afterwards, a random target flip-flop is chosen where the fault is injected at a random time. The SEU in a flip-flop can be emulated in the simulator by inverting the value of the target flip-flop. This procedure is automated by using a standard state-of-the-art simulator and its routines and commands. In the presence of a fault, the simulation run is compared with the golden run and any differences at the output or any differences in the functional behaviour of the circuit can be observed and recorded.

IV. Reliability Analysis of Secured BNN Hardware Inference Engine

A. Implementation of Power-Side Channel Masking in BNNs

As mentioned in Section II.C, several power side-channel masking techniques are proposed in the state of the art, e.g. [8]. In our analysis, we focus on the weighted summations as the most critical to soft-error reliability according to our hypothesis.

Instead of an adder tree, which is used for the popcount implementation that may be vulnerable to power side-channel attack [5], a ripple-carry style adder is recommended. For the normal 1-bit RCA, the logic functions for the summation and the carry-out bit are provided in (3) (4), respectively, where a, b, c, C and S are the two one-bit inputs, one-bit carry-in, carry-out and summation.

$$S = a \oplus b \oplus c; \tag{3}$$

$$C = a \cdot b \oplus b \cdot c \oplus a \cdot c; \tag{4}$$

Based on these equations, the complete addition computations are expressed as a sequence of XOR and AND operations. As indicated in [27], only the AND operation in the summation needs to be masked since it is a non-linear operation, while the XOR operation is a linear operation and, therefore, can be left unmasked. Among the recent related masking styles [28], the Trichina method is chosen for AND gate masking because of its simplicity and efficiency [8]. However, the straightforward adoption of Trichina's AND gate causes glitches in the circuit [29], which can lead to information leakage. Thus additional registers are introduced to stabilize the signals in the design. The glitch-resistant Trichina's AND gate is shown in Fig. 2, where the secret variables (a and b) are split into different shares (a_0, a_1, b_0, b_1), r is a fresh random bit, and the additional registers are marked in red. The equations used for sum bits and carry bits are shown respectively in (5), (6), (7) and (8)).

$$S_0 = a_0 \oplus b_0 \oplus c_0; \tag{5}$$
$$S_1 = a_1 \oplus b_1 \oplus c_1; \tag{6}$$
$$C_0 = Tri_0(a_0, b_0, r_0) \oplus Tri_1(b_0, c_0, r_1) \oplus Tri_2(c_0, a_0, r_2) \tag{7}$$
$$C_1 = Tri_0(a_1, b_1, r_1) \oplus Tri_1(b_1, c_1, r_1) \oplus Tri_2(c_1, a_1, r_2) \tag{8}$$

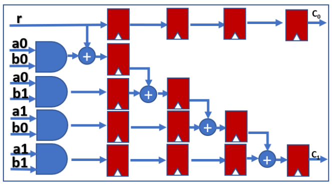


Fig. 2: Trichina AND gate

Fig. 3 shows the 1-bit ripple-carry style security-masked adder with the Trichina AND gate used in the RCA.

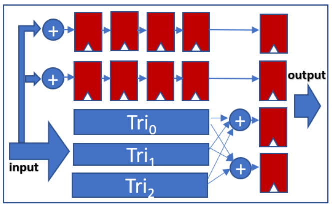


Fig. 3: 1-bit security-masked adder

For the final 1-bit security-masked adder, 12 more registers (marked in red) complement the Trichina AND gate to address the introduced delay in the Trichina AND gate. In total, for 1-bit security-masked adder, 54 ($14 \cdot 3 + 12 = 54$) registers are added. Generally, an N-bit ripple carry adder is formed by N 1-bit full adders, as shown in Fig. 4. Therefore, for one N-bit security-masked adder, the total number of registers introduced to the design is $54 \cdot N$. Depending on the adder's size N and the number of summation resources instantiated in the BNN, a significant number of registers may be introduced that increase the susceptibility of the design to soft-errors.

B. Security-Masked BNN-Based Case Study Design

In order to study how the weighted summation masking technique influences the soft-error reliability of the BNN

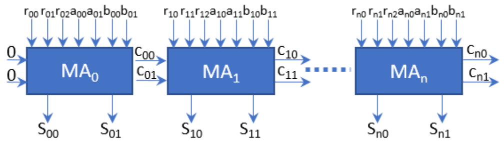

r00 r01 r02 a00 a01 b00 b01
r10 r11 r12 a10 a11 b10 b11
rn0 rn1 rn2 an0 an1 bn0 bn1
0
0
MA0
C00
C01
MA1
C10
C11
MAn
Cn0
Cn1
S00
S01
S10
S11
Sn0
Sn1

Fig. 4: N-bit Masked Adder

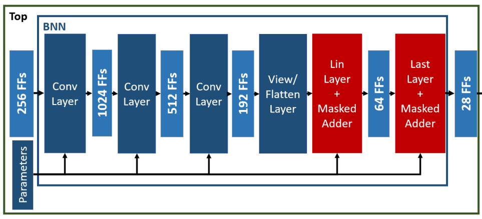


Fig. 5: A Case-study security-masked BNN design

hardware implementation, we apply the above masking technique on a case-study ultra-low power near-sensor binarized neural network implementation proposed in [30]. There are six layers in total which include three convolutional layers, one flattening layer and two fully connected layers. The DPA countermeasure target masking technique is initially applied to the fully connected layers [8] and the last two layers of the BNN hardware implementation influence the output the most, therefore we apply the masking technique to the last two fully connected layers in the BNN. We replace the weighted summation operation (i.e. popcount) with the N-bit security-masked adder explained in section IV-A: 8-bit security-masked adder and 7-bit security-masked adder for layers 5 and 6 separately. Due to the number of summation resources, the security-masked design introduces around 5 million flip-flops. The final design is shown in the Fig. 5 and the security-masked layers are marked in red.

C. Reliability Assessment Setup

Replacing an original adder with the security-masked adder introduces many additional flip-flops to the circuit. Each additional flip-flop, in turn, increases the probability of SEU faults. However, soft-error effects can be masked as discussed in Section III. To evaluate the actual reliability jeopardy by the added flip-flops and to compare the resulting failure rate of the original BNN with the security-masked BNN design, the de-rating factors have to be computed.

The case-study BNN implementation [30] provides a functional testbench which applies 100 different input data samples (images) to the network and the related outputs are categorised into 4 classes. For the experiment, the testbench is modified in the way that all bit-wise changes of the BNN outputs as well as the predicted class (i.e. the functional result) are monitored and recorded for each applied input. This simulation serves as the golden run.

The faults, i.e. inverted values of the target flip-flops, are injected at a random target and at a random clock cycle during the active workload of the simulation. Thus, each simulation run is independent and several runs can be performed in parallel. Since the simulation time of the original (i.e. non-masked) BNN design is rather short it is possible to run a complete fault injection campaign. This means that for each target flip-flop enough fault injection simulations are performed to cover every clock cycle during the active stimuli of the testbench. For the modified security-masked case-study BNN design (i.e. with the implemented DPA countermeasures) the number of targets and the simulation time are significantly higher and therefore, the random fault injection sampling approach has to be used.

For each simulation run, the output of the circuit is monitored and any difference to the golden run is traced as an output failure. Additionally, the predicted class is compared to the golden simulation and the predicted class altered a functional failure is traced.

To obtain the de-rating factors for the output or functional failure, the number of observed failures is divided by the total number of injected faults. The failure rate is then calculated using Equation (2). Since the FIT rate of the flip-flops depends on the applied nano-scale implementation technology and the analysis in this paper is performed on a higher level, for both designs, a normalized FIT rate of $\mathrm{FIT}_{\mathrm{FF,\,SEU}} = 1$ is assumed for each flip-flop. Equation (2) is simplified to multiply the number of flip-flops with the de-rating factor to obtain the final failure rate $\mathrm{SER}_{\mathrm{SEU}}$ of the design.

V. Experimental Results

A. *LDR Analysis of the Masked Full Adder*

In order to analyse the critical parts of the BNN, an evaluation of the logical de-rating is performed on the (security-masked) ripple carry adder. The analysis is performed with the SoCFIT tool by IROC, as described in section III-B. Different sizes (N) of the adder are considered and the results for the maximum and average LDR are shown in Table I.

TABLE I: LDR Analysis of the (Masked) Full Adder

	Fulle Adder		Masked Full Adder	
Size N (bits)	LDR (avg)	LDR (max)	LDR (avg)	LDR (max)
4	0.61	1.0	0.98	1.0
8	0.57	1.0	0.99	1.0
16	0.55	1.0	0.99	1.0

The maximum LDR is 1.0 for both adders and unaffected by the adder size N. This means, considering the worst case every fault in one of the sequential cells of the adder is propagating to the output of the adder. The average LDR, however, is decreasing for the normal full adder and increasing for the masked full adder. Generally, the LDR for the full adder is lower than the LDR for the masked full adder. This means that a fault in any of the sequential cells of the masked full adder is more likely to propagate to the output than a fault in the conventional full adder.

The used LDR analysis approach is pessimistic because it shows an upper bound for the LDR of the cell, without using any information of the actual workload. Thus, LDR is the probability for a fault affecting an internal cell to propagate to the primary outputs of the circuit. It does not include any aspects related to the criticality of the fault. Additionally, the BNN consists of different stages which contain other functional blocks besides the adder. An exhaustive analysis should therefore, consider the adder in the full context. This is achieved by performing an exhaustive fault-injection simulation campaign of the full BNN/masked BNN design. This campaign determines the Functional De-Rating factors and is presented in the next section.

B. *Fault-Injection Simulation Campaign*

This section presents the results obtained from the performed fault injection simulation campaigns. As mentioned in the previous section, the simulation time of the masked BNN design is significantly higher and thus, a full fault injection campaign over the entire available test data was not feasible. A random sampling fault injection campaign was performed instead and in order to confirm that enough fault injection simulations have been sampled, several simulation campaigns with a varying number of fault injections were executed. Fig. 6 shows the de-rating factors for the output and functional failure measured from these campaigns depending on the number of injected faults. It can be seen that that the de-rating factors are converging and the carried out fault injection simulations were sufficient.

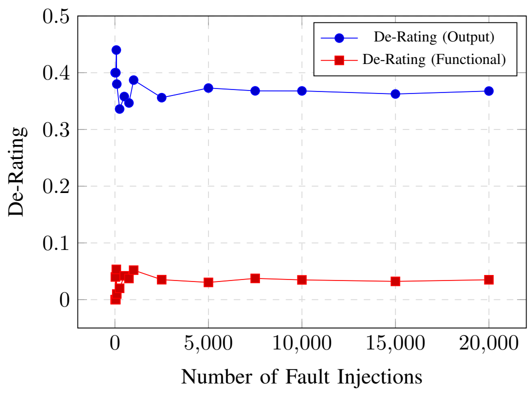


Fig. 6: Convergence of the masked BNN de-rating factors values in the random sampling fault-injection campaign

TABLE II: Fault Injection Results

	BNN	Masked BNN
Flip-Flops	2076	5 348 912
Number of Injections	197 220	62 760
Output Failures	188 186	23 000
De-Rating (Output)	0.95	0.37
Failure Rate (Output)	1980.91	1 960 245.00
Functional Failures	48 948	2165
De-Rating (Functional)	0.25	0.03
Failure Rate (Functional)	515.24	184 518.71

All the simulation campaigns shown in Fig. 6 run a different subset of fault injections (different target flip-flops with different input data) and therefore, can be accumulated to obtain the overall de-rating and failure rate of the masked case-study BNN design. The results are summarized in Tab. II together with the results of the full fault injection campaign of the original case-study BNN design.

Table II provides a comparison of the the original BNN and the masked BNN designs with regard to their expected reliability. Although the output and functional de-rating are about $3\times$ and $10\times$ lower for the masked BNN, due to the considerable high amount of additional flip-flops, the resulting failure rate is about $1000\times$ higher for the output and $350\times$ higher for the functional failures. This proves that the studied masking techniques used in the literature for power side-channel countermeasures significantly increase vulnerability of the HW BNN inference engines to soft errors.

VI. Conclusions

Hardware BNN inference engines are gaining their popularity in the computer engineering domain and stamp their inevitable presence in the security and reliability critical applications. However, the efforts of security and reliability R&D communities happen to be fragmented to address the issues specific to their domains. A prominent example is a power side-channel countermeasure based on a set of masking techniques introduced to address a recently discovered security vulnerability. The approach overlooks potential issues for reliability and creates a significant vulnerability to radiation-induced single event effect faults, i.e. soft errors, in the field.

This paper has presented an analysis for the soft-error reliability jeopardy by the DPA side-channel mitigation measures in hardware implementations of BNN inference engines. The work reveals reliability issues, i.e. a steep increase of vulnerability to single-event effects, introduced by the security enhancement techniques, and emphasizes the interdependency of the design's reliability and security aspects. The analysis demonstrates that due to the considerable number of additional flip-flops established by the security countermeasure and their chained position, the bit-wise failure rate is about $1000\times$ higher for the BNN output and $350\times$ higher for the neural network functional failures.

As the future work, we plan to study options for reliability-aware side-channel countermeasures for security enhancement in DNN inference engines, i.e. aiming at cross-aspect optimization solutions.

References

[1] V. Sze *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *Pro. of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[2] J. Zhang *et al.*, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 159–172.

[3] M. Courbariaux *et al.*, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[4] M. Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Computer Vision – ECCV 2016*, B. Leibe *et al.*, Eds., 2016.

[5] A. Dubey *et al.*, "Maskednet: The first hardware inference engine aiming power side-channel protection," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 197–208.

[6] P. Kocher *et al.*, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[7] E. Brier *et al.*, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.

[8] A. Dubey *et al.*, "Bomanet: boolean masking of an entire neural network," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[9] M. A. Hanif *et al.*, "Robust machine learning systems: Reliability and security for deep neural networks," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 257–260.

[10] T. Simons *et al.*, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

[11] R. N. Reith *et al.*, "Efficiently stealing your machine learning models," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, ser. WPES'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 198–210. [Online]. Available: https://doi.org/10.1145/3338498.3358646

[12] L. Batina *et al.*, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 515–532. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/batina

[13] H. Yu *et al.*, "Deepem: Deep neural networks model recovery through em side-channel information leakage," *2020 IEEE Int. Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 209–218, 2020.

[14] E. Trichina, "Combinational logic design for aes subbyte transformation on masked data." *IACR Cryptol. EPrint Arch.*, vol. 2003, p. 236, 2003.

[15] G. Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[16] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*. Springer, Boston, MA, 2011.

[17] D. Alexandrescu *et al.*, "Towards optimized functional evaluation of see-induced failures in complex designs," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, 2012, pp. 182–187.

[18] S. Mittal, "A survey on modeling and improving reliability of dnn algorithms and accelerators," *Journal of Systems Architecture*, vol. 104, p. 101689, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762119304965

[19] G. Abich *et al.*, "Soft error reliability assessment of neural networks on resource-constrained iot devices," in *2020 27th IEEE Int. Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2020, pp. 1–4.

[20] Y. Ibrahim *et al.*, "Soft errors in dnn accelerators: A comprehensive review," *Microelectronics Reliability*, vol. 115, p. 113969, 2020.

[21] Y. He *et al.*, "Fidelity: Efficient resilience analysis framework for deep learning accelerators," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 270–281.

[22] W. Li *et al.*, "Soft error mitigation for deep convolution neural network on fpga accelerators," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 1–5.

[23] B. Salami *et al.*, "On the resilience of rtl nn accelerators: Fault characterization and mitigation," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2018, pp. 322–329.

[24] A. Ruospo *et al.*, "Evaluating convolutional neural networks reliability depending on their data representation," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 672–679.

[25] M. A. Neggaz *et al.*, "Are cnns reliable enough for critical applications? an exploratory study," *IEEE Design Test*, vol. 37, no. 2, pp. 76–83, 2020.

[26] L.-H. Hoang *et al.*, "Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *Proc. Design, Automation and Test in Europe*, ser. DATE '20. San Jose, CA, USA: EDA Consortium, 2020, p. 1241–1246.

[27] E. Trichina *et al.*, "Small size, low power, side channel-immune aes coprocessor: design and synthesis results," in *International Conference on Advanced Encryption Standard*. Springer, 2004, pp. 113–127.

[28] O. Reparaz *et al.*, "Additively homomorphic ring-lwe masking," in *Post-Quantum Cryptography*. Springer, 2016, pp. 233–244.

[29] S. Mangard *et al.*, "Successfully attacking masked aes hardware implementations," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, J. R. Rao *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 157–171.

[30] M. Rusci *et al.*, "Design automation for binarized neural networks: A quantum leap opportunity?" *CoRR*, vol. abs/1712.01743, 2017. [Online]. Available: http://arxiv.org/abs/1712.01743

Curriculum Vitae

1. Personal data

Name	Xinhui Lai
Date and place of birth	May 28^{th} 1990, Henan Province, China
Nationality	Chinese

2. Contact information

Address	Tallinn University of Technology, School of Information Technologies, Department of Computer System, Ehitajate tee 5, 19086 Tallinn, Estonia
Phone	+372 58488565
E-mail	xinhui.lai@taltech.ee

3. Education

2017–present	Tallinn University of Technology, School of Information Technologies, Department of Computer System, PhD studies
2014–2017	Politecnical di Torino, Faculty of Electronic Engineering, Embedded System, MSc
2010–2014	Politecnical di Torino, Faculty of Electronic Engineering, Electronic and Communications Engineering, BSc

4. Language competence

Chinese	Native
English	Fluent

5. Professional employment

2017–Now	Tallinn University of Technology, Early Stage Researcher
2017–2021	MSCA ITN "RESCUE" Early Stage Researcher, Host Tallinn University of Technology, Estonia
2022–Now	Nokia Corporation, Specialist SoC/IP Design & Verification

6. Computer skills

- Operating systems: Uinux, Windows
- Document preparation: Microsoft office, Latex
- Programming languages: VHDL/Verilog, Python, Tcl, C

7. **Honours and awards**

   - 2014–2017 EDISU scholarship

8. **Defended theses**

   - 2014, Hardware Implementation of Polar Encoder, MSc, Supervisor Prof. Guido Masera, Politecnical di Torino, Faculty of Electronic Engineering

9. **Field of research**

   - Hardware security
   - Extra-functional aspects verification
   - Design verification

Elulookirjeldus

1. Isikuandmed

Nimi	Xinhui Lai
Sünniaeg ja -koht	28.05.1990, Henan Provints, Hiina
Kodakondsus	Hiina

2. Kontaktandmed

Aadress	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutisüsteemide instituut, Akadeemia tee 15a, 12618 Tallinn, Estonia
Telefon	+372 58488565
E-post	xinhui.lai@taltech.ee

3. Haridus

2017–2022	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutisüsteemid, doktoriõpe
2014–2017	Politecnical di Torino, Faculty of Electronic Engineering, Embedded System, magistratuur
2010–2014	Politecnical di Torino, Faculty of Electronic Engineering, Electronic and Communications Engineering, bakalaureus

4. Keelteoskus

Hiina keel	Emakeel
Inglise keel	Kõrgtase

5. Teenistuskäik

2017–Praegu	Tallinna Tehnikaülikool, nooremteadur
2017–2021	MSCA ITN "RESCUE" nooremteadur, Tallinna Tehnikaülikool
2022–Praegu	Nokia Corporation, SoC/IP projekteerimise & verifitseerimise spetsialist

6. Arvutialased oskused

- Operatsioonisüsteemid: Uinux, Windows
- Tekstitöötlus: Microsoft office, Latex
- Programmeerimiskeeled: VHDL/Verilog, Python, Tcl, C

7. Autasud

- 2014–2017 EDISU stipendium

8. Kaitstud lõputööd

- 2014, Hardware Implementation of Polar Encoder, magistritöö, juhendaja Prof. Guido Masera, Politecnical di Torino, Faculty of Electronic Engineering

9. Teadustöö põhisuunad

- Riistvara turvalisus
- Ekstra-funktsionaalsete aspektide verifitseerimine
- Funktsionaalne verifitseerimine

ISSN 2585-6901 (PDF)
ISBN 978-9949-83-852-3 (PDF)