Dipl.-Ing. Armin Krieg

Control State Based Fault Detection and Analysis in Multi-Processor Systems-On-Chip

Dissertation

vorgelegt an der
Technischen Universität Graz

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften
(Dr. techn.)

durchgeführt am Institut für Technische Informatik
Technische Universität Graz
Vorstand: Em. Univ.-Prof. Dipl.-Ing. Dr. techn. Reinhold Weiß

Graz, im Januar 2013

Kurzfassung

In den vergangenen Jahren gab es einen dramatischen Anstieg der Systemintegrationsdichte von integrierten Halbleiterimplementierungen. Diese Marktentwicklung wurde in erster Linie von zwei Faktoren getrieben. Erstens, der Markt für Unterhaltungselektronik forderte eine wachsende Menge an integrierter Funktionalität in mobilen Anwendungen. Zweitens, neue Fertigungstechnologien im deep-submicron Bereich erlaubten diese steigende Integration bei nahezu gleichbleibendem Energiebedarf. Ein negativer Nebeneffekt dieser hohen Integrationsdichten ist ein breites Spektrum an Zuverlässigkeitsproblemen, die von den Systemdesignern gelöst werden müssen. Weiters müssen auch absichtliche Fehler berücksichtigt werden, welche durch einen Angreifer in das System eingebracht werden um dieses in einen fehlerhaften Zustand zu bringen. Während es in den letzten Jahren ein breites Spektrum an Forschung auf dem Gebiet von Untersuchungstechniken in frühen Designständen gab, z.B. simulations- oder emulationsbasierte Ansätze, gab es kaum Bewegung am Gebiet der Fehlererkennung.

In dieser Arbeit werden neue Methoden der effizienten kontrollflussbasierten Laufzeitprüfung der Ausführungsintegrität gesucht und evaluiert. Zu diesem Zwecke wurden neue signatur-basierte Techniken eingeführt, welche eine Verschmelzung der Fehlerdetektions- mit einer Leistungsabschätzunginfrastruktur ermöglichen. Auf Grund der stark eingeschränkten Resourcen in der gewählten Zieldomäne werden weiter neue Methodiken zur Abschätzung der Fehlererkennungs-, Resourceverbrauchs-, Stromverbrauchseffizienz benötigt. Diese Eigenschaften sollen mit Hilfe einer FPGA-basierten Emulationsplatform evaluiert werden. Daher wurden im Rahmen dieser Arbeit neue Methodiken entwickelt, um schnelle und genaue Evaluierungen mittels solcher Emulationssysteme zu ermöglichen. Die hierfür implementierten Fallstudien basieren auf einer von Gaisler Research entwickelten SPARC-v8 Implementierung (LEON3). Weiters wurde die generelle Anwendbarkeit der vorgestellten Techniken in einem industriellen Anwendungsbeispiel demonstriert, welches auf einem echten Smart-Card System inklusive einem sicheren Betriebssystem basiert.

Abstract

Recent years showed a dramatic increase in system integration density of embedded semiconductor implementations. This development was driven by two main factors. First, market demand for ever more functionality residing in small mobile devices. Second, deep-submicron semiconductor manufacturing processes allowed to increase density at a steady power consumption. High integration density, on the other side, results into a variety of additional problems for the system designers. A major concern in recent and future designs in this context is execution reliability caused by operational faults. Another fault source that got into the focus of various research groups are intentional ones caused by malicious attackers, trying to put the target system into an unintended state. While there has been a wide range of research concerning early design phase investigation techniques, like simulation or emulation approaches, detection has only slightly improved.

In this work novel techniques are explored to enable efficient on-line checking of operation integrity. For this task novel signature-based techniques are introduced, which enable the integration of fault detection mechanisms into power estimation infrastructure. Also, such new implementations require methodologies to test their effectiveness in terms of fault detection, resource usage, and power consumption. These operational properties are evaluated using two FPGA-based emulation systems, one of academic nature and an industrially used one. Therefore, this work introduces new techniques to enable fast and accurate system evaluations using general emulation-based evaluation platforms. The provided case-studies are based on the open available SPARC-v8 implementation provided by Gaisler Research (LEON3). Furthermore, an industrial case study using a smart-card hardware implementation and secure operating system is presented to prove the general applicability of the introduced approaches.

Acknowledgements

This thesis has been pursued as part of the POWER-MODES research project, which is a successful cooperation between Infineon Technologies Austria AG, AustriaCard GmbH and the Institute for Technical Informatics of Graz University of Technology. Considering all these organizations and their employees that supported me during the accomplishment of my thesis, I would like to thank them all, but of course I am only able to mention a few.

First of all, I would like to thank Prof. Reinhold Weiss for providing this possibility to work in the hardware/software co-design research group at the Institute for Technical Informatics. Especially, I would thank for the helpful comments and advice given during the review phases of this work. Furthermore, I would like to express my gratitude to Christian Steger for his thoughtful supervision of the project and support concerning the academic publishing process.

Our industry partners provided us with continuing insight into state-of-the-art products and industrial design environments. In this regard I would like to thank Josef Haid for deep insight into idea creation processes and his knowledge concerning low power design and power estimation. Also, I would like to give my thanks to Holger Bock for his efficient project management and his deep knowledge of smart-card security. Furthermore, my thanks are due to Christiane Ulbricht and Thomas Aichinger for sharing their experiences with smart-card operating systems.

Special thanks go to my PhD colleagues Norbert Druml and Manuel Menghin for their in-depth review of this work and their helpful comments on this thesis. Also, I would like to thank Christian Bachmann and Andreas Genser for their comprehensive insight into power emulation and their various tips that were a tremendous help during my PhD. In this regard I would also like to give my thanks to my project partner Johannes Grinschgl for the great cooperation during the accomplishment of the POWER-MODES project.

Finally, I would like to thank my family for their continuing support and especially my dear Sisi for her enduring understanding even when this thesis consumed most of my free time. Without you I would not have been able to finish this work. Thank you!

Graz, January 2013 Armin Krieg

Extended Abstract

In 1965 Gordon Moore of the Intel Cooperation postulated that the amount of transistors integrated into semiconductor systems will double every two years. His assumptions stand true up until now and, according to Intel, will hold for the many years to come. This development was enabled by a continuous improvement of semiconductor manufacturing processes and system integration techniques. Currently, deep sub-micron processes enable structures of 22nm and smaller sizes, resulting in an isolation thickness of only a few atomic layers. From a system perspective, dense *Systems-on-Chip* (SoCs) implementations concentrated many different functions (processor cores, audio and video IP cores, and communication interfaces to name a few) into single packages.

This highly increased integration density came at a price: high peak power consumption caused by the high number of sub-systems; dependability issues and device degradation problems caused by the fragility of deep sub-micron structures; security issues caused by the intermixture of personal data and entertainment functions. These challenges have to be dealt with at the earliest possible design-stage to avoid problems after market entry of the finalized product. Hardware emulation approaches, based on *field programmable gate arrays* (FPGAs), aim to solve previously described challenges at once. First, direct synthesis of the RTL description of the final product for FPGA evaluation platforms, provides functional equivalence of the evaluation and production hardware model. Second, the hardware implemented model is enabled to be evaluated at realistic clock-rates, improving speed and timing equivalence (depending on system complexity). These advantages are used in hardware-accelerated software verification platforms, already supporting the hardware/software co-design process for years. Unfortunately, these verification systems do not provide power information or allow the investigation under fault conditions.

The POWER-MODES project[1], which included and financed this work, aims at providing a comprehensive evaluation platform for security, dependability, and power investigations. This inter-disciplinary approach merges solutions for different but highly connected problem spaces into a single hardware-accelerated evaluation system. While the focus lies on the high performance verification of software implementations, the derived techniques are also applicable on highly focused hardware verification tasks. This way, an efficient possibility for *hardware/software codesign* is created for the estimation of system power efficiency and dependability under fault conditions. An overview over such a comprehensive approach, combining two different domains into a single evaluation platform, is depicted in Figure 1.

[1]*POWer EmulatoR and MOdel based DEpendability and Security evaluation platform*, collaborative research project of the Graz University of Technology, Infineon Austria AG and AustriaCard GmbH. Funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 825749.

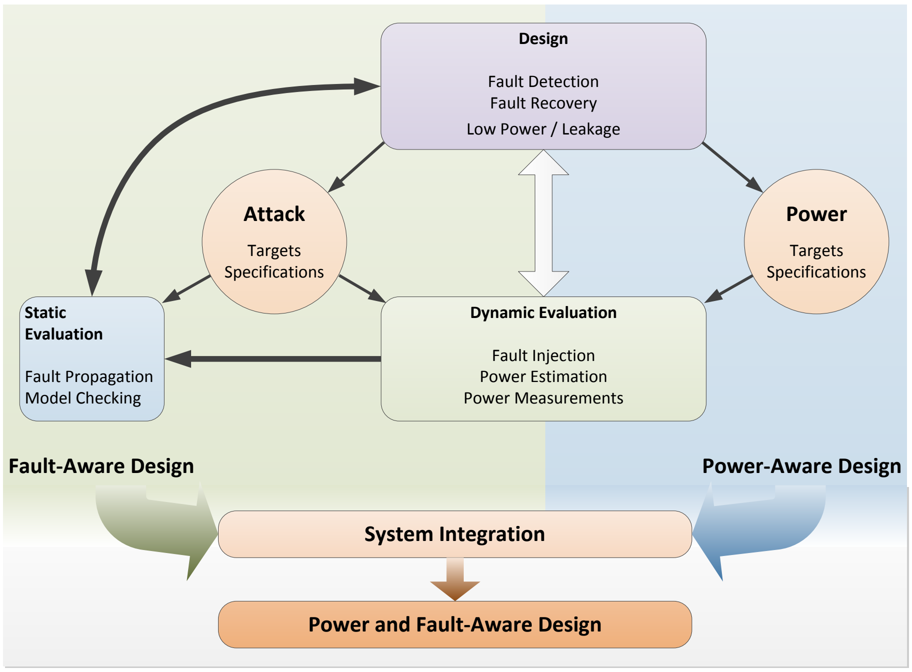


Figure 1: Control-flow-based Online Fault Detection and Analysis - A Comprehensive Approach

As visualized in Figure 1, power-aware and fault-aware design relies on techniques from various backgrounds. This thesis aims at providing a novel methodology to evaluate existing system implementations, using static and dynamic techniques, and to design variants that solve identified weaknesses. First, the power-aware design approach for the development of embedded software has to be extended for enabling *power analysis* techniques used in security evaluations. For this purpose, a hardware-accelerated methodology has been presented, introducing such functionality into a proven power characterization and evaluation flow[2]. Such strategies can be categorized as *dynamic evaluation* techniques.

Second, to enable the merger of security-aware power estimation, as discussed earlier, and direct fault-attack investigations into a single evaluation platform, additional *fault injection* circuitry is necessary. In the course of this work, a modular implementation of such fault injection hardware has been derived and integrated into existing software verification infrastructure[3]. The experiences gained during the creation of a data-aware power estimation platform has been used to introduce novel countermeasures against such forms of attack. Under the consideration of the challenges posed by the design of resource-constrained systems, only existing infrastructure has been reused to keep additional hardware effort as low as possible[4].

[2] *Krieg et al., Accelerating Early design Phase Differential Power Analysis Using Power Emulation Techniques*, 4th IEEE International Symposium on Hardware-Oriented Security and Trust 2011 (HOST '11), San Diego, United States of America, 5-6th of June 2011.

[3] *Grinschgl et al., Case Study on Multiple Fault Dependability and Security Evaluations*, Elsevier Microprocessors and Microsystems (MICPRO), accepted for publication, 2013.

[4] *Krieg et al., A Side Channel Attack Countermeasure using System-On-Chip Power Profile Scrambling*, 17th IEEE Intl. On-Line Testing Symposium 2011 (IOLTS '11), Athens, Greece, 13-15th of July 2011.

All previously discussed evaluation techniques and design practices have been integrated and investigated inside a single platform. Furthermore, hardware security attacks, as shown in previous literature, have been mapped onto this system and successfully presented[5]. Large multi-processor system-on-chip (MPSoC) implementations introduced various new issues concerning security and dependability. Under the consideration of these new challenges, also the power estimation approach was modularized and data from the physical implementation was included to increase the evaluation accuracy[6]. Again, advances in the field of evaluation techniques can be used to improve various design methodologies. The heterogeneous nature of MPSoCs poses a tremendous challenge for secure system design. Considering that system-wide fault detection has to be provided, a micro-architectural signature approach has therefore been introduced to cover this gap in existing literature[7]. The integration of such novel techniques into existing implementation is a major concern in the design of large systems. Therefore, an automatized approach has been shown to generate complete monitor modules using the previously presented signature characterization process[8].

The major drive behind all introduced techniques for evaluation and design of digital systems shown in this work, has been security-aware engineering. Safety engineering suffers from many similar problems, with the difference that faults are caused by random sources[9]. The dynamic evaluation of large heterogeneous digital systems is increasingly limited by available investigation techniques and resources. Hence, formal verification methods are introduced in various parts of the verification process. Furthermore, static analysis of the hardware description allows for the linking of the high-level RTL view and the low-level physical implementation of the design[10].

[5] *Krieg et al., POWER-MODES - POWer EmulatoR and MOdel based Dependability and Security evaluations*, ACM Transactions on Reconfigurable Technology and Systems (TRETS), Volume 5, Issue 4, Article 19, 2012

[6] *Krieg et al., System Side-Channel Leakage Emulation for HW/SW Security Coverification of MPSoCs*, 15th IEEE Symposium on Design and Diagnostics of Circuits and Systems 2012 (DDECS '12), Tallinn, Estonia, 18-20th of April 2012.

[7] *Krieg et al., Characterization and handling of Low-Cost Micro-Architectural Signatures in MPSoCs*, 17th IEEE European Test Symposium 2012 (ETS '12), Annecy, France, 28th of May to 1st of July 2012.

[8] *Krieg et al., PROCOMON - An Automatically Generated Predictive Control Signal Monitor*, 15th Euromicro Conference on Digital System Design 2012 (DSD '12), Izmir, Turkey, 5-8th of September 2012.

[9] *Krieg et al., Power and Fault Emulation For Software Verification and System Stability Testing in Safety Critical Environments*, IEEE Transactions on Industrial Informatics (TII), to be published in Volume 9, Issue 2, 2013.

[10] *Krieg et al., Acceleration of Fault Attack Emulation by Consideration of Fault Propagation*, 11th Intl. Conference on Field-Programmable Technology 2012 (FPT '12), Seoul, Korea, 10-12th of December 2012.

Contents

List of Figures

List of Tables

List of Abbreviations

ACM	Association for Computing Machinery
AES	Advanced Encryption Standard
API	Application Programming Interface
ASIP	Application-Specific Instruction Processor
BDD	Binary Decision Diagram
CAN	Controller Area Network
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic-Redundancy-Check
DES	Data Encryption Standard
FPGA	Field Programmable Gate Array
FSM	Finite-State Machine
HDL	Hardware Description Language
HW	Hardware
IDE	Integrated Development Environment
IP	Intellectual Property
IEEE	Institute of Electrical and Electronics Engineers
ITRS	International Technology Roadmap for Semiconductors
LUT	Look-Up Table
MBU	Multi-Bit Upset
MPSoC	Multi-Processor System-on-Chip
NoC	Network-on-Chip
NVM	Non-Volatile Memory
PE	Power Emulation
RFID	Radio Frequency Identification
RAM	Random-Access Memory
ROM	Read Only Memory
RSA	Rivest-Shamir-Adleman Public Key Cryptography
RTL	Register Transfer Level
SER	Soft Error Reliability
SEU	Single Event Upset
SMV	Symbolic Model Verifier
SoC	System-on-Chip
SMP	Symmetric Multi-Processing
SPARC	Scalable Processor Architecture
SPICE	Simulation Program with Integrated Circuit Emphasis
SW	Software
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WCET	Worst-Case Execution Time

Glossary

Control State Based Fault Detection
The term *control state based fault detection* has been artificially formed from the commonly used concept *control flow checking* [15, 16] and the concept of *processor fingerprinting* as described in [17]. This technique uses checkpoint signatures of the current processor state, hence, such hashes allow to check if the system behaves correctly compared to a given reference. The latter principle is not generic enough for the purpose of this work, as it is only applicable to modular-replicated processors running in lock-step mode or by comparing with long pre-recorded signatures.

Multiprocessor Systems-on-Chip (MPSoC)
The term *multiprocessor systems-on-chip* (MPSoC) has been used in recent years to describe complex heterogeneous chip implementations combining the advantages of data processing parallelism of *multi-processors* and high-level integration of *systems-on-chip* [18, 19]. The large scale introduction of these multi-functional devices enabled many functionalities common in today's smart-phones (i.e. multi-protocol communication, audio and video codecs etc.).

Efficiency
The term *efficiency* has to be substantiated depending on the field it is applied to. Concerning fault detection, *detection efficiency* specifies the ability to detect a fault during run-time. *Power efficiency* describes a system's capability to regulate its power consumption and to utilize available resources in an effective way. If system performance is concerned, *efficiency* means the ability to execute a given program using available resources in an effective manner.

Information Leakage
The term *information leakage* means the dependency of the power profile on processed data. This direct or indirect dependency could lead to the leakage of internal information to the outside, if an external person applies profile-processing techniques.

Fault
A *fault* constitutes a deviation of normal internal system states or signals. Such deviation could lead to the generation of wrong results, but it could also be masked by the current system state.

Error
An *error* describes a deviation from the expected system behavior caused by a fault. Therefore, an error is a final consequence after a fault was activated and the result is stored by internal or external resources.

Deep-submicron process technology
Commonly the expression *deep-submicron process technology* is used for manufacturing processes using transistor channel lengths smaller than 0.18um. Such a distinction was necessary as scaling below this size resulted in increasing problems concerning process variability, leakage power and signal integrity.

Chapter 1

Introduction

1.1 Motivation

As of today, since the introduction of the latest smart-phones, we are all living in a mobile world. Regardless of our current local position, we are able to communicate with everyone who is also connected to one of the many open communication channels. This was mostly made possible by the semiconductor design, verification, and manufacturing industry that enabled the ongoing validity of moore's law.

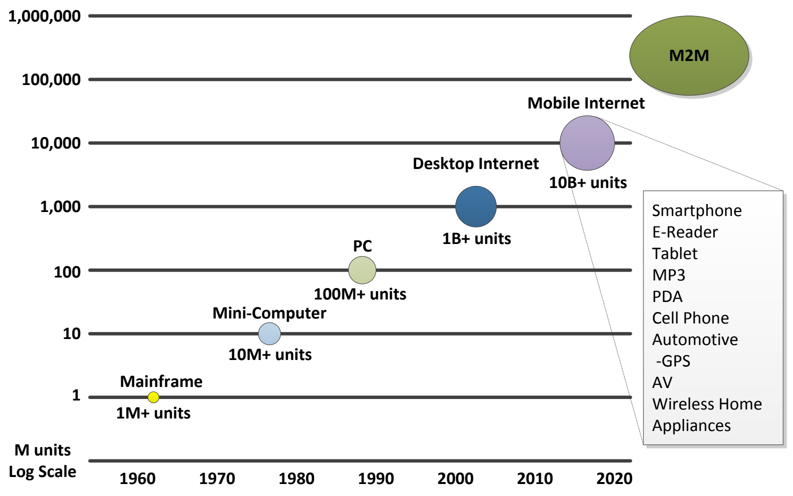


Figure 1.1: Dawn of the Machine-to-Machine (M2M) Age (adapted from [1]) - the development of the mobile device market is far from over, M2M communication will further increase the number of mobile devices sold

Billions of mobile devices are already sold world-wide, with increasing numbers every year. The now gradual introduction of IPv6[1] will boost these markets even more in the near future, enabling *machine-to-machine* (M2M) communication (see Figure 1.1).

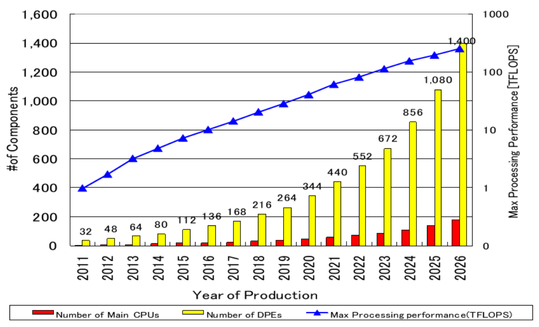
This development introduced a variety of new challenges for system designer, especially

[1] *Internet Protocol version 6*, extending the addressing capability of the Internet protocol version 4 significantly to 2^{128} devices

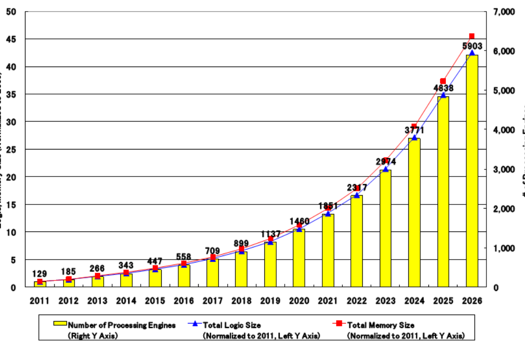
concerning power-aware and fault-aware design. First, while research made a continuing increase in integration density of integrated circuits possible, mobile power sources improved much slower. This development gap made the introduction of various power estimation (during design) and management (during operation) techniques necessary to efficiently use finite power sources. Second, deep sub-micron semiconductor processing technologies, necessary to integrate billions of transistors in small energy efficient devices, increasingly reach the physical limits of atomic barriers. Besides increasing manufacturing and testing challenges, long-term reliability is a concern in modern devices suffering from device degradation effects like *negative bias temperature instability* (NBTI). Third, a variety of mobile devices is now involved in the processing of critical personal data. In 2012, over seven billion smart secure devices will be shipped, with annual growth rates of up to 20 percent [20]. These gigantic markets and the increasing complexity of the involved devices call for ever improving hardware and software co-design, verification, and test support.

1.1.1 David and Goliath - Design Challenges of the Large and Tiny

Recent years showed an increasing usage of specialized hardware and multi-core processors. This development was driven by the need for higher operation performance while classic energy sources did not develop as quickly. Also the introduction of the *green computing* idea, made the design of highly energy-aware systems for data-centers necessary. In case of system-on-chip implementations, the semiconductor industry publishes forecasts concerning design and market challenges. In Figure 1.2 such developments of the *international technology roadmap for semiconductors* (ITRS) concerning system heterogeneity and complexity are depicted. Figure 1.2a shows the non-linear growth in system components during the following years. It can be seen that the usage of classic *central processing units* (CPUs) is not growing as strongly as *dedicated processing elements* (DPE). The reason for that is that dedicated hardware can solve specific problems much more efficiently than generalized hardware can. This is true for both, operating performance and energy efficiency.



(a) Heterogeneity for large stationary computing systems [2]



(b) The complexity of mobile system-on-chip designs [2]

Figure 1.2: Market developments in stationary and portable systems concerning complexity, performance, and implementation effort, according to the ITRS

In Figure 1.2b the complexity of future mobile-systems is presented. Here, it is clearly visible that such portable system-on-chips increase non-linearly in complexity, while energy sources are only improving slowly. Therefore, every part of such complex system-in-chips has to be optimized for their power consumption. To guarantee a certain level of reliability of such systems, including their large in-chip communication networks like *network-on-chips* (NoCs), system-level fault detection and management mechanisms have to be provided.

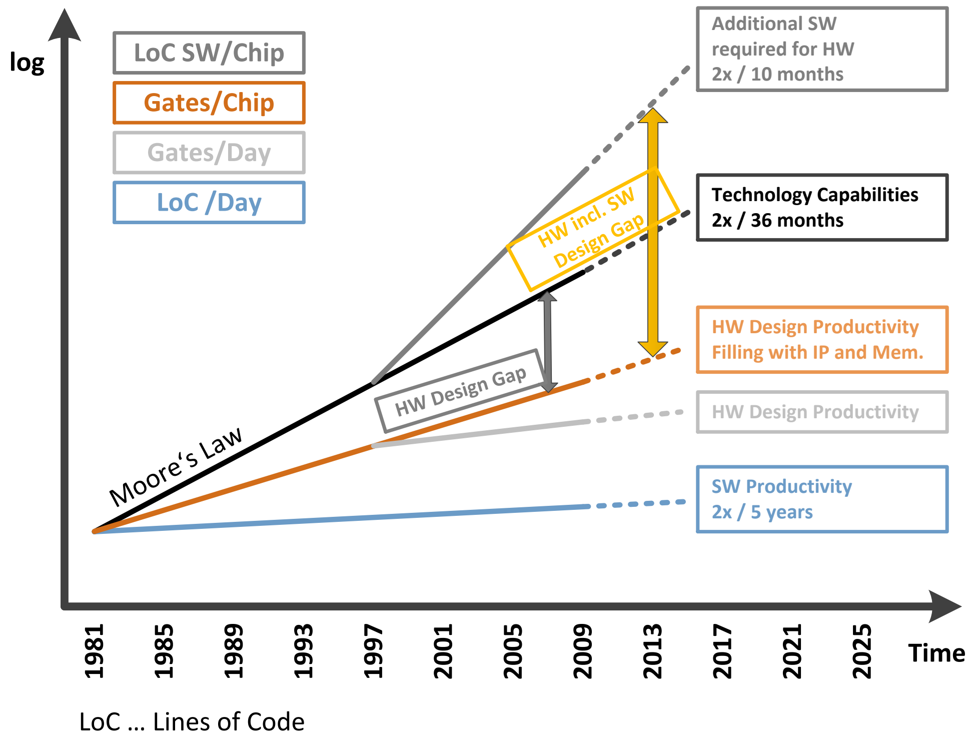
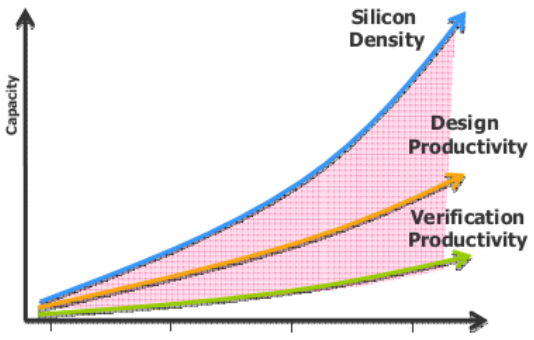
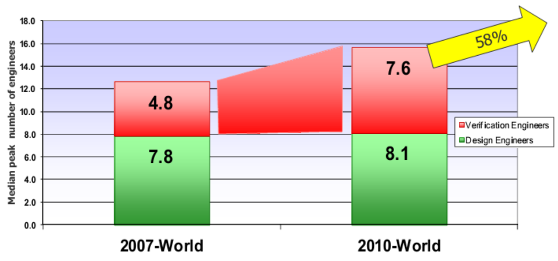


Figure 1.3: The gap between software, hardware design productivity, and required firmware is continuously widening if no countermeasures are taken in time (adapted from [2])

The semiconductor and system industries also have to solve various challenges concerning the software/hardware co-design process. In Figure 1.3 a disturbing development in recent years is depicted, that will worsen in future years if no solutions are found. In this figure, it can be seen that while technology capabilities (according to Moore's law) double every 36 months, hardware design productivity improved much slower. Only concerning hardware development, this challenge has been partially mastered by reusing *intellectual property* (IP) blocks and memory-based techniques. On the other hand, software productivity evolves even slower and, at the same time, these complex systems rely on ever more software to control this hardware (doubling every 10 months). This leads to a strongly opening shear between the demand for software and hardware, and the ability to provide these. Only interdisciplinary approaches, for example the introduction of software product line engineering and the wider usage of system-describing languages that can be compiled into hardware, will be able to reduce this challenge.



(a) The famous verification gap revisited [3]



(b) Verification effort in recent years [3]

Figure 1.4: Challenges concerning the verification of future complex systems according to Mentor Graphics

Another urgent challenge in the implementation of large heterogeneous system-on-chips concerns test and verification. Already in 1999, the SEMATECH industry consortium published data concerning the so called ”verification gap”, meaning the increasing challenge to verify large systems. In 2011, Mentor Graphics showed that this gap is still, even after the growing introduction of formal methods, widening as shown in Figure 1.4a. This situation led to an increasing demand for verification engineers while the demand for design personnel was stagnating (visualized in Figure 1.4b). This increase in verification effort was mostly driven by functional verification, hence, the validation of requirements concerning power consumption and fault-robustness have to be supported in the coming years to counteract the widening of this verification gap.

1.1.2 Power- and Fault-Awareness in Modern System-On-Chip Designs

As described in previous paragraphs, the complexity of modern SoC implementations is increasing tremendously and power-awareness has already been a strong challenge in recent years. One major step towards system energy efficiency was the stop to the frequency race and the large scale introduction of parallelism into consumer products. Integrated cores or functional units can be completely deactivated during run-time if they are not currently needed. This was made possible by advances in recent years concerning techniques like *clock and power gating* as shown in [21] and *dynamic voltage and frequency scaling* (DVFS) as described in [22]. These techniques all work on a global scale, meaning they reduce consumption by manipulating architectural contributers.

The recent increase in implementation density has been mostly driven by the continuing shrinkage of semiconductor technology nodes. While small transistor sizes have a positive effect on power consumption (except leakage power, which is dramatically worsening if no manufacturing techniques like *silicon-on-insulator* (SOI) are used) and available chip space, thinner isolation and shorter transistor channels introduced tremendous reliability problems. Besides faster device aging, manufacturing variability is affected by the demands of modern deep sub-micron processes. To emphasize how strongly this factor is worsening, current data and a forecast for future process nodes is depicted in Figure 1.5.

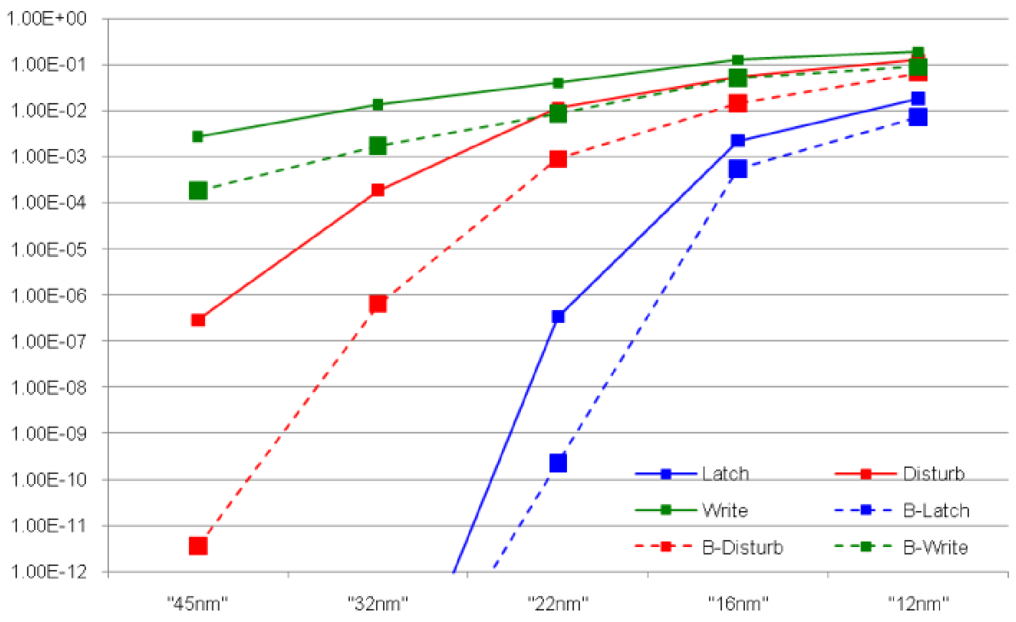


Figure 1.5: Manufacturing variability is strongly worsening for deep sub-micron technologies, according to [2]

This development led to a wide variety of research concerning fault detection, recovery, and fault-robust devices such as [23]. However, on a system-level duplication is still the first choice for high-reliability implementations, specifically triple-module-redundancy. Therefore, novel strategies for such system-level techniques, that do not rely on duplication, and the evaluation concerning their power-awareness, are strongly needed.

1.1.3 Hardware-Accelerated Emulation for Software Verification

Traditional simulation techniques to verify software/hardware co-designs are increasingly suffering from performance bottlenecks and therefore, limit their usability in the design flow. This situation led to the increasing usage of hardware assisted emulation or FPGA prototyping as depicted in Figure 1.6.

These techniques can be divided into several subgroups depending on their application in the design process and the system abstraction. For this work we will identify two major fields:

- Mapping of hardware implementations onto hardware accelerated evaluation platforms and systems. Depending on the size and abstraction of the system implementation, single FPGAs or large reconfigurable logic clusters are used. This approach has the advantage that even finalized netlists can be verified for their correctness without any influence of the applied test hardware. Otherwise, large FPGA clusters can only provide investigations at low clock rates. There is an additional problem of trust, if such platforms should be provided to external parties, for example software providers.

- System or hardware descriptions are extended with additional structures to allow for internal monitoring or manipulation during run-time. The design is directly optimized for the target platform, therefore it is not completely equivalent to a real final implementation. The main advantage is the strict separation between hardware and software using defined interfaces. Therefore, such evaluation platforms can also be provided to external parties when the used netlists are encrypted.

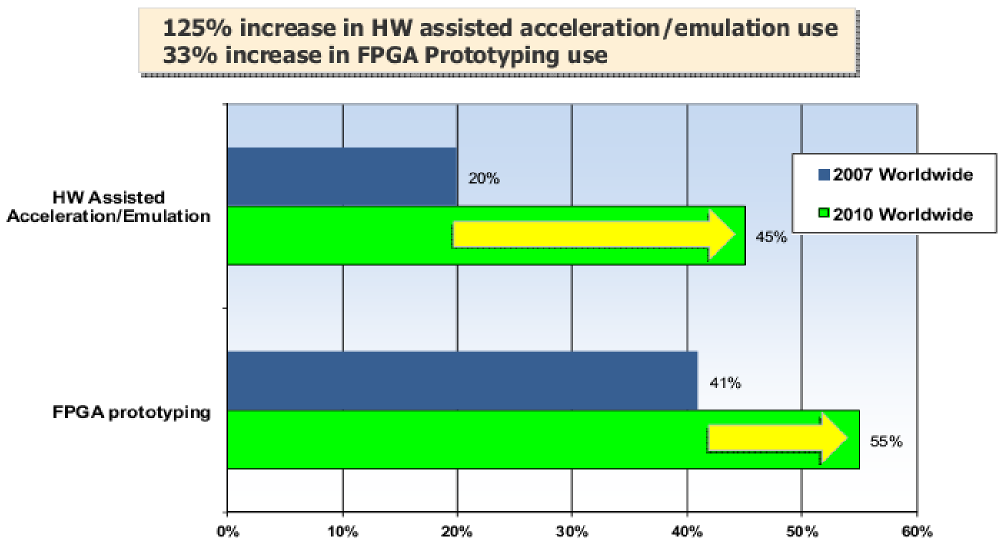


Figure 1.6: Usage trends for hardware-accelerated emulation during system design and verification, according to [3]

While the first field has been researched very thoroughly in recent years (for example [24]), instrumented verification platforms for software/hardware co-verification have only been covering selective implementation challenges. For example instrumented emulation for fault testing as shown in [25], software verification for power consumption as presented in [26], or variability emulation as introduced in [27]. To the best of our knowledge, there is no comprehensive approach providing all this information inside one evaluation platform, also considering challenges of trust and security available in literature.

1.2 Control-State-Based Fault Detection and Analysis in Multi-Processor System-On-Chips

1.2.1 The POWER-MODES Project

This thesis is part of the "**POW**er **E**mulato**R** and **MO**del based **DE**pendability and **S**ecurity evaluation platform" (POWER-MODES) collaborative research project[2], shared between the Institute for Technical Informatics at the Graz University of Technology, Infineon Technologies Austria AG, and AustriaCard GmbH. This combined effort targets the combination of early power estimation and fault injection techniques into a single software verification platform as depicted in Figure 1.7.

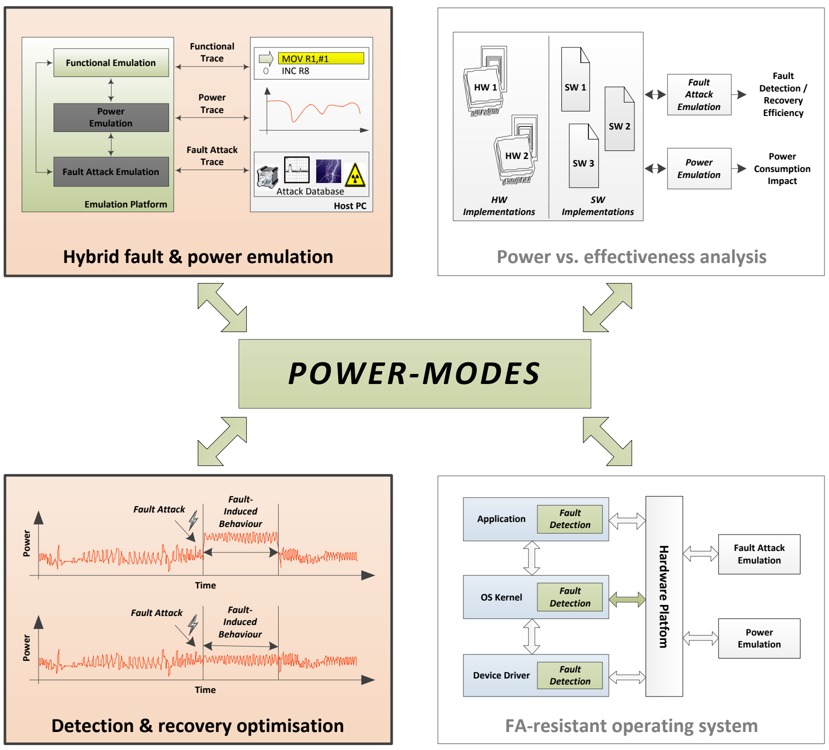


Figure 1.7: The POWER-MODES project - Tackling power-aware and fault-aware software evaluations concurrently using a single hardware-accelerated emulation platform (adapted from [4])

The POWER-MODES approach covers four main fields of research: *hybrid fault and power emulation*, *optimization of fault detection and recovery techniques*, *comparative analysis of power-effectiveness versus detection effectiveness*, and *fault-attack resistant operating system, design*. This work will concentrate on the first two topics, enabling a hybrid emulation system combined with the exploration of novel detection and recovery techniques.

[2]Funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 825749.

1.2.2 Problem Statement

Large heterogeneous system-on-chips as well as resource-constrained systems used in security-critical and safety-critical applications demand significant optimization and verification effort. State-of-the-art hardware/software co-design flows have not been developed with these corner cases in mind and therefore, the following deficiencies can be identified:

- Traditional fault detection techniques provide local detection
- Power efficiency of fault detection mechanisms are scarcely researched
- Lack of early evaluation techniques for the testing of power-awareness and fault-awareness
- Lack of information-leakage estimation during secure software verification
- Simulation-based investigation flows demand high computational effort
- Lack of static analysis methods leads to a large evaluation space and missing knowledge about the system-internal process during faults

These limitations have been addressed by the combination of existing hardware accelerated evaluation techniques and novel estimation and detection methodologies. Signature-based mechanisms are introduced to provide system-level fault detection capability that is highly scalable depending on detection and power requirements. Early evaluation of power consumption and information leakage during the software verification process is tackled by a novel data-aware emulation methodology. Hence, the presented signature-based detection mechanisms can also be evaluated for their power consumption effort.

This thesis also introduces automatized techniques to integrate these new detection and evaluation mechanisms into system-on-chips and FPGA-based software verification platforms. The problem of investigation complexity is addressed by novel static analysis techniques. Furthermore, the similarity of design challenges in the security and safety domains have been considered.

1.2.3 Contributions and Significance

In summary, this thesis provides contributions to the following fields:

1. *System-Level Control-State-Based Fault Detection*: A fully automated process analysis hardware behavior for a given set of benchmark applications. The introduced methodology allows for the selection of significant control signals to derive signatures of the system state. Resource-constrained systems are supported by a hardware characterization process to determine the signature generation quality of reduced hash implementations. Finally, control-state information is utilized to automatically generate hardware checker modules for run-time integrity checking. This novel approach enables signature-based on-line testing for both, large heterogeneous systems and resource-constrained designs.

2. *Emulation-Based Analysis of Fault Induced Security and Safety Issues*: The emulation framework, as introduced in this thesis, allows the concurrent evaluation of power-related, security-related, and safety-related issues in a single platform. The hardware-accelerated approach uses an abstracted programming interface to allow the direct integration of the presented techniques into standardized software verification systems. This novel combined methodology has been further extended with data-aware power sensors to enable the early estimation of information leakage for a given leakage intensity. The presented investigation techniques have been thoroughly tested using software implementations of the AES algorithm and safety self-test routines.

1.2.4 Structure of the Work

The remaining parts of this work are organized as follows. In Chapter 2 the current state-of-the-art concerning power-estimation and fault-injection in hardware/software co-design methodologies is reviewed. This complex thematic field first has to be split into hardware-accelerated and simulation-based power-estimation for the early investigation of a program's power consumption. Second, FPGA-based fault-injection approaches are investigated, considering both security and dependability fields. Third, the static and dynamic evaluation of software and hardware, as part of the co-design process, is reviewed to connect all three investigation parts into a single verification system. Chapter 3 presents a novel methodology for the efficient evaluation of secure software implementations under fault conditions. Part of this approach is also the resource-saving integration of novel control-flow-based fault detection mechanisms into MPSoCs. Therefore, these techniques cover the complete hardware/software co-design process, from system evaluation to novel design possibilities to counteract detected deficiencies. Chapter 4 provides data from comprehensive evaluations that show the applicability of the previously introduced techniques. These results are generated using open available implementations of system-on-chip code bases to show the generality of the approach. In Chapter 5 the results of this thesis are concluded summarizing novel insights gained into secure system designs. Furthermore, an outlook into future work in this dynamic field of engineering is given. Finally, Chapter 6 presents a selection of publications on which this work is based. These give a deeper understanding of the implemented techniques and additional evaluation data generated during the POWER-MODES project.

Chapter 2

Related Work

The comprehensive power-aware and fault-aware hardware/software co-design process presented in this work is built on three basic pillars: *hardware-accelerated software verification techniques*, *fault detection in secure and dependable system-on-chips*, and *static system evaluation*. These topics have been partially covered by many publications in recent years, improving the state-of-the-art for selective problems. Only both, the dynamic and static evaluation of the system-under-test can provide global coverage of the system security and dependability problem. The following section gives an overview about existing work in these three categories. Finally, a summary is given showing how this work improves the state-of-the-art in these fields, and how they are connected into a comprehensive approach.

2.1 Hardware-Accelerated Dynamic Verification Techniques

Dynamic verification of hardware and software has been researched for many years with different evaluation targets. Here, we will concentrate on three main topics: *emulation-based power estimation* to cover the power-awareness field, *power analysis techniques for software verification* including all early applicable measures to analyze the power consumption profile for information leakage, and *fault injection-based system analysis* covering approaches to test systems under the influence of operational faults. Concerning power estimation and characterization of computing devices, a thorough overview is given in [28].

2.1.1 Emulation-Based Power Estimation

An overview on the basic principles of the power emulation technique is shown in [29]. This hardware-accelerated methodology provides run-time improvements of factor 10 to 500 compared to simulation-based power estimation approaches. Also strategies to minimize the hardware overhead caused by the emulation methodology are given in the presented work. This approach has been extended by the authors of [30] into a hybrid power estimation methodology for complex SoCs. The use of combined simulation and emulation techniques led to significantly reduced power consumption investigation times. The possibility of direct interpretation is a main advantage of hardware-assisted run-time power estimation. This has been exploited for the power-aware process migration between cores as shown in [31]. A real-time power profiling unit that can be directly integrated into FPGA-based software evaluation platforms has been presented in [26].

To allow for high estimation accuracy and low preparation effort, an automated system-on-chip characterization methodology has been introduced in [32]. The application of these novel early power estimation techniques for software power profiling has been further investigated in [33]. Another application is the evaluation of *dynamic voltage and frequency scaling* (DVFS) strategies as introduced in [34].

2.1.2 Power Analysis Techniques for Software Verification

Depending on the abstraction level, information leakage can be estimated using a variety of different techniques. First, a purely software-based solution could be applied as introduced in [35]. Side channel leakage information is extracted from a given program by execution and tracing of instruction and processor states. Extensible analysis module are then used to process the resulting information. These analysis modules are directly implemented for the Microsoft Debugger API, integrating the functionality into the *integrated development environment* (IDE). Unfortunately, no information about accuracy and simulation performance has been given by the authors.

Second, instruction set simulators can provide in-system knowledge for power consumption estimation. Although, high simulation speeds can be reached using this approach, it is depending on data only available for thoroughly profiled general purpose architectures. Such methodologies have been presented in [36] and [37]. While not giving any hints about the accuracy of the used power profiles, both solutions promise high simulation speeds. In a case of badly estimated information leakage, a software engineer could be led into a sense of false security if this leakage is higher than anticipated. The work shown in [37] relies on several abstract power models to isolate possible sources of information leakage. As described earlier, well characterized general purposes processors are necessary for this approach. In this case, the AVR microcontroller series of Atmel has been chosen. Based on a performance simulator a comprehensive approach for power, area, and timing modeling has been taken in [38].

Third, highly accurate gate-level or even transistor-level simulations can be applied to identify critical parts of a system-under-test. High costs in terms of simulation speed and needed simulation equipment are the price of such estimation accuracy. Therefore, a system designer has to decide whether to choose between very short simulation times or the isolated evaluation to the smallest possible leakage contributer. As it is of utmost importance to know if an adversary could extract information using a high number of power traces, the first choice can be considered infeasible. Highly accurate analog simulations using SPICE simulators have been used in the following work: [39]. *register transfer level* (RTL) simulations as presented in [40] can be also used to improve simulation speed. Furthermore, it has been noted by the authors of [40] that RTL simulations could hide information leakage that will only be visible after later design flow steps.

2.1.3 Fault Injection-Based System Analysis

The reliability of high-safety applications under the influence of faults is an important feature for which testing methods using fault injection have been introduced [41]. Depending on the design stage, faults can be injected into finalized parts e.g., using radiation, manipulation, or during earlier phases using adaptable prototyping approaches. These methodologies can be further divided into simulation-based and emulation-based techniques depending on the application of the evaluated models.

High-level hardware descriptions offer the possibility to be simulated directly during early design phases. This fact has been exploited by first non-invasive fault injection tools using logic simulation. During design concept phases, it could also be of interest to apply such injection methods using system-level description languages like SystemC as shown in [42]. Advances to simulation-based techniques have been introduced for the VFIT tool in [43]. Compared to previous work, the authors complemented this approaches with automatized saboteur and mutant insertion techniques. Furthermore, various injection performance improvements of simulation and emulation approaches have been shown in [44]. Recent work utilizes existing scan-chain infrastructure for high-performance processor reliability evaluations [45].

Higher fault injection coverage and therefore, higher evaluation performance could only be achieved by the introduction of hardware-accelerated emulation techniques. Such emulation methodologies promise significantly higher injection rates as presented in [25]. Therefore, a higher amount of possible fault configurations could be evaluated than using traditional simulation-based approaches. The introduction of novel FPGA devices featuring the possibility of run-time reconfiguration, without influencing the data- and control-flow, opened new possibilities for non-invasive fault injection. Since the introduction of these techniques several improvements concerning flexibility and performance have been introduced [24].

These approaches have been developed for dependability investigations, targeting mostly *single event upsets* (SEU). *Multiple event upsets* (MEU) need more complex emulation techniques, such as the extensive use of run-time reconfiguration as described in [24]. Such fault constellations are especially important for security evaluations as intentional faults could be injected at several positions at once. The complexity of today's security and safety systems often results in the separation of software and hardware development, done by different entities. This introduces a conflict of trust and intellectual property that has not been considered by previous work described in literature.

2.2 Fault Detection in Secure and Dependable Systems

The detection of operational faults during the execution of critical software sequences has been a very active field of research for many decades. Such faults can be detected and managed on three different abstraction levels. In this work, we will concentrate on signature-based techniques: only on software-level using *software-based signature mechanisms*, on hardware-level using *hardware-based signature mechanisms*, and comprehensive techniques exploiting *hardware-software co-design solutions.*

2.2.1 Software-Based Signature Mechanisms

Signature-based approaches on the software-level mostly rely on the automatic or semi-automatic generation of source-code signatures. These can then be used to detect changes in the program flow caused by adversarial tampering or environmental influences. Such signatures are directly embedded into the executed binaries, so for the checking procedure itself processor internal resources have to be reused. This will result in a severe increase of program run-time.

First software solutions have been relying on the usage of extensive redundancy. Another strategy would be the application of control-flow state checking to identify unforeseen execution behavior as introduced in [46] and [47]. A pure software signature-based approach, generating these signatures during compile time is proposed in [16]. These software-only approaches come with the big disadvantage of having a significant impact on execution performance (40 - 600% performance and memory overhead). Additionally, an adversary could manipulate these software checks if such techniques are implemented in security critical systems.

2.2.2 Hardware-Based Signature Mechanisms

Hardware monitors have been the most commonly used hardware blocks for integrity checking for many decades. While monitoring capability can be implemented very efficiently depending on the monitored unit, this advantage could be annulled if pre-computed signatures are stored in large memories. Furthermore, the selection of the monitored system region is a not well covered field in the current state-of-the-art. For system-on-chip design's heterogeneous nature approaches that only cover a processor's pipeline are insufficient. The use of multiple monitors increases system complexity, decreases area efficiency and therefore, is also not well suited.

As the impact of the monitoring process on the execution performance has to be reduced, direct access to hardware resources is needed. Therefore, control flow-monitoring implementations using dedicated monitoring hardware have been presented. Early work concerning dedicated monitoring hardware have been watchdog-type modules such as [48] have been introduced. Another possibility would be the integration of smaller specialized monitoring circuits covering only selected parts of the system as described in [49] and [50]. While these techniques use an external view of the target, also approaches relying on a modification of a processor pipeline itself have been shown to enable the on-line evaluation of the control flow. As stated earlier these techniques are very effective when monitoring only processor applications, but are not sufficient for heterogeneous applications like system-on-chips that include various different processing units.

2.2.3 Hardware/Software Co-Design Solutions

In recent years, various hardware/software co-design based approaches have been published to provide wide detection coverage. These techniques exploit the possibility to adapt both software support and hardware structure to reduce memory overhead and performance degradation. Predestined for such systems are *application-specific instruction processors* (ASIP) as described in several recent papers [51]. However, such methodologies can only be applied if the target architecture is either highly adaptable or the design process is at a very early stage allowing for strong interventions in existing development flows.

If such design flow changes are not possible, it is also possible to extend existing hardware to generate representative values to track control changes. For this purpose test infrastructure like scan-out-chains can be exploited like the approach presented in [52]. While the impact on the complexity of the targeted system is small, it relies on periodic tests and therefore, cannot be considered a real on-line testing solution. The same research group presented fingerprinting techniques based on hash-values generated from the architectural state [17]. This very accurate fault detection mechanism leads to high demands on circuit bandwidth because of the large amount of data collected from system components. Further improvements and extensions to this technique for large chip multi-processors have been presented in [53].

2.3 Static System Evaluation

The increasing complexity of system-on-chip designs becomes a growing challenge for test and verification engineers. Therefore, static system information has to be utilized to decrease the evaluation space or to verify the implementation in a formal manner. This led to the increasing usage of formal methods in various parts of the design process: *high-level verification for security and safety domains*, *classic application of formal methods in hardware verification*, *static VHDL code analysis*, and finally, *fault propagation modeling.*

2.3.1 High-Level Verification for Security and Safety Domains

The implications of *loss of trust* (security) or *loss of life* (safety) led to the introduction of strong regulatory bodies in the security and safety engineering domains. In recent years, researchers in the safety engineering field spent a significant research effort on the verification of software implementations concerning their correctness and standard compliance. Model checking is a formal verification technique with the goal to determine whether a *finite state machine* (FSM) satisfies a set of specified criteria under all circumstances. If the criterion is not fulfilled, a counterexample for debug support is generated. This technique proved to be a powerful tool for such tasks. A comprehensive approach, using the NuSMV model checker including the consideration of fault injection, has been introduced in [54]. Nonetheless, the presented work is limited to software implementations and classic verification challenges like liveness or reachability properties. An efficient application of model checking to verify fault tolerance by using mutated models of sub-systems has been shown in [55]. For the application of formal methods to a secure software implementation flow, principal concepts have been introduced in [56, 57]. As abstraction is a challenge concerning RTL hardware descriptions, the presented techniques cannot be directly applied to hardware certification.

2.3.2 Classic Application of Formal Methods in Hardware Verification

Disadvantages of dynamic verification can be counteracted by the application of static formal techniques to prove the correctness of a circuit using mathematical methods. These do not depend on signals, clocks, waveforms, or test patterns and therefore, do not suffer from large exploration spaces. For this work, we will concentrate on the model checking technique, as it has been shown to work well for the automatized verification of software implementations.

Model checkers operate on two inputs: the model $\mathcal{M}$, denoted by FSM transitions and the systems properties ϕ that should be fulfilled, denoted by formulas in temporal logic [58]. Models $\mathcal{M}$ have to be abstracted manually as such checkers can only verify models in their own language, this manual task limits the applicability of current implementations. Therefore, much research effort has been spent to enable the automated extraction of such control flow elements from RTL hardware descriptions. Full verification tools incorporating both, FSM extraction and symbolic model checking, have been introduced in various early work (see Section 6.9). Besides being a proprietary implementation, these works were mainly targeting basic implementation challenges, like reachability analysis, without considering checks for high-level rules. Similarly, earlier work describes the transformation of simple synchronous circuits into FSMs. An automated approach for the abstraction of hardware descriptions written using the Verilog language has been introduced in [59]. These approaches mainly focus on bit-level evaluations and therefore, are not well scalable for very large system-on-chips. From this situation resulted the introduction of techniques like *Predicate Abstraction*, or *Term-Level Abstraction*. Further improvements have been shown in [60], applying *high-level decision diagrams* (HLDDs) not only to describe a system's behavior but also to correct wrong node transitions in an automated manner. In this work, an open source implementation of a symbolic model checker from the *new symbolic model verifier* (NuSMV) project is used.

2.3.3 Static VHDL code analysis

The evaluation of hard real-time and safety systems is highly depending on timing and particularly *worst-case execution time* (WCET) analysis. In the past, such investigations have been based on hand-crafted timing models extracted from a formal specification using VHDL or Verilog. As this error-prone process results in significant engineering effort, a framework for the static analysis of VHDL code has been introduced in [61]. The authors treat VHDL descriptions like sequential programs to derive a control flow graph in CRL2 (a specially developed language for the description of control flow graphs). An abstraction-aware compiler for such hardware description models has been introduced by the same group in [62]. The simulation of the non-deterministic view of the underlying architecture is enabled by the extraction of C++-code from the hardware description. Finally, these techniques and semi-automatic processes of hardware model timing analyses have been combined into the framework presented in [63]. As the focus of this work has been on safety and real-time applications (especially the modeling of timing behavior) in the security domain, besides control flow also information flow has to be evaluated. Consequently, a state-machine-based approach solving this challenge has been introduced in [64] to determine information flow at a higher abstraction level. This high abstraction level has been established to reduce impractical large evaluation spaces resulting from low-level logic investigation. This research group extended the approach further for the introduction of a novel hardware description language targeting a secure information flow [65].

2.3.4 Fault Propagation Modeling

Low-level logic evaluation for *soft-error reliability* (SER) could only be done using slow but very accurate SPICE-based simulations until recent years. Various new methodologies based on *binary decision diagrams* (BDD) to improve such investigations performance-wise have been introduced in [66]. Thus enabling the optimization of the tested circuits to improve this reliability metric as well as the early estimation of SER on a logic-level. Furthermore, SER and logic masking in combinational and sequential circuits have been analyzed using a signature-based approach as described in [67]. In the security domain the work proposed in [68] is of special interest as it specifically targets multiple transient faults as they would be expected during complex fault-attacks. Unfortunately, is not directly applicable to security applications, although it describes extensive fault models and dependability metrics. These topics have only been partially covered in [69] using AES implementations as an example.

2.4 Summary and Difference to the State-of-the-Art

The early estimation of power consumption is of high importance during the design of complex hardware/software co-designs. Emulation-based approaches are a promising methodology to do such early estimations at a high evaluation speed even before silicon is available. Another important reliability property of embedded systems is the robustness to faults caused by environmental effects or injected by an adversary. Fault injection using manipulations of a system's hardware description or by direct run-time reconfiguration of an FPGA have been well proven techniques to test these properties. Currently, no combined evaluation of power and robustness metrics is available. Furthermore, the separated development of software and hardware by different entities has not been extensively considered in literature.

Detection of operational faults during the execution of safety-critical and security-critical software is a well covered field of research. Traditional techniques include the extensive use of software and hardware replication or the usage of fault detection and correction codes. These techniques work well if the system complexity is limited or if there are no power and cost constraints. In large system-on-chips or resource-constrained implementations, such methodologies could impose unacceptable cost penalties or uncovered system-regions.

Static analysis of software implementations has a long tradition in safety-critical system design. On the other hand, formal techniques for hardware verification only cover small parts of hardware implementations, like reachability analysis, testing for lifeness properties, or equivalency checking. The verification of high-level requirements, as given by formal descriptions demanded by regulatory bodies, has got only very limited coverage by current literature. Especially considering the ever increasing complexity of system-on-chip designs, static analysis methods are strongly needed in the hardware/software co-design process.

This thesis aims at introducing the following main improvements to the state-of-the-art, with respect to the goals defined in Section 1.2.3:

- Introduction of a hardware-accelerated software-verification platform for the testing of secure software implementations under the consideration of separated software/hardware development model
- Implementation of system-wide fault detection mechanisms for power-aware resource-constrained systems using automatized generation techniques

Furthermore, this work introduces the following auxiliary advances:

- Extension of previous work on emulation-based power-estimation to enable the evaluation of information leakage depending on a given leakage level
- Exploration of static system evaluation techniques to enrich the certification-aware hardware verification process with formal methods
- Exploration of static evaluation of RTL hardware descriptions to allow the automatized generation of support hardware for hardware-accelerated investigation platforms

Chapter 3

Fault Detection and Analysis for Multi-Processor Systems-on-Chip

3.1 Overview

Fault analysis using fault injection methods on various abstraction levels has become an inevitable step during the design and verification of reliable and secure systems. The proposed techniques aim at supporting this process from the design of robust system-level detection mechanisms over hardware-accelerated verification platforms to static analysis methods for the evaluation of fault-attack effects. The various contributions, which are part of a comprehensive approach, are opposed to the individual topics in Figure 3.1.

The basis of such an evaluation and design methodology is a comprehensive FPGA-based investigation platform. The work presented in Section 6.1 extends existing *power emulation* techniques with data-aware functionality to support *differential power analysis* used during security testing. Pure emulation-based *fault injection* techniques are shown in 6.2 including an industrial case study to prove the applicability of the approach. The application of such an investigation platform for the design and test of a novel *power profile scrambling* technique is described in Section 6.3. This approach can be further extended for the *fault attack* robustness evaluation of cryptographic software and hardware implementations, like AES (see Section 6.4). Multi-processor systems-on-chip pose a significant challenge during security evaluation of hardware/software co-designs because of their heterogeneous nature. Therefore, a sensor-based technique for accurate *information leakage* estimation using physical implementation data has been introduced in Section 6.5. Building on such a multi-functional verification and test platform, novel *control based fault detection* mechanism can be evaluated for their efficiency (see Section 6.6). This technique, using *micro-architectural signatures*, has been further extended with automatized hardware generation to enable the creation of memory-less hardware checker modules (see Section 6.7). As security and safety domains are concerned regarding similar operational reliability challenges, the concurrent evaluation of power- and fault-robustness properties of safety-critical software implementations is presented in Section 6.8. The fast growing complexity of systems-on-chip and the problem *fault propagation* resulted into the need for extensive static code analysis as introduced in Section 6.9.

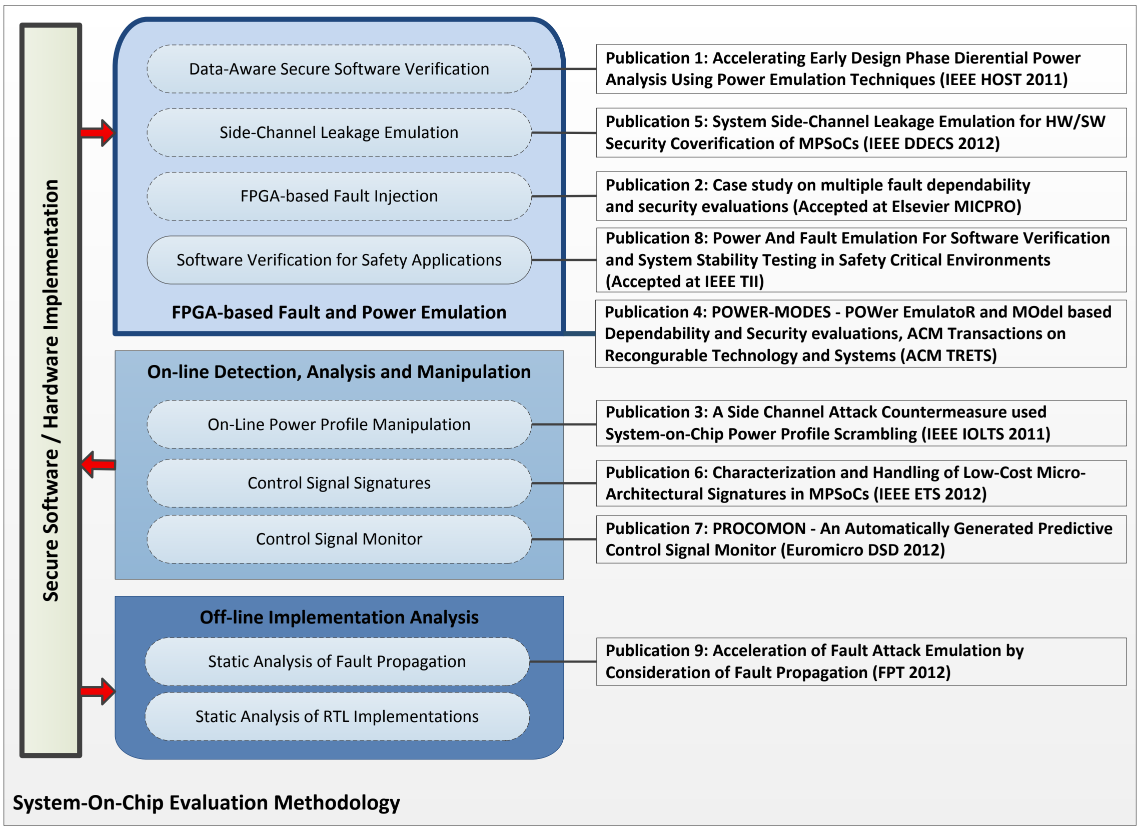

Secure Software / Hardware Implementation
Data-Aware Secure Software Verification
Side-Channel Leakage Emulation
FPGA-based Fault Injection
Software Verification for Safety Applications
FPGA-based Fault and Power Emulation
On-line Detection, Analysis and Manipulation
On-Line Power Profile Manipulation
Control Signal Signatures
Control Signal Monitor
Off-line Implementation Analysis
Static Analysis of Fault Propagation
Static Analysis of RTL Implementations
Publication 1: Accelerating Early Design Phase Dierential Power Analysis Using Power Emulation Techniques (IEEE HOST 2011)
Publication 5: System Side-Channel Leakage Emulation for HW/SW Security Coverification of MPSoCs (IEEE DDECS 2012)
Publication 2: Case study on multiple fault dependability and security evaluations (Accepted at Elsevier MICPRO)
Publication 8: Power And Fault Emulation For Software Verification and System Stability Testing in Safety Critical Environments (Accepted at IEEE TII)
Publication 4: POWER-MODES - POWer EmulatoR and MOdel based Dependability and Security evaluations, ACM Transactions on Recongurable Technology and Systems (ACM TRETS)
Publication 3: A Side Channel Attack Countermeasure used System-on-Chip Power Profile Scrambling (IEEE IOLTS 2011)
Publication 6: Characterization and Handling of Low-Cost Micro-Architectural Signatures in MPSoCs (IEEE ETS 2012)
Publication 7: PROCOMON - An Automatically Generated Predictive Control Signal Monitor (Euromicro DSD 2012)
Publication 9: Acceleration of Fault Attack Emulation by Consideration of Fault Propagation (FPT 2012)
System-On-Chip Evaluation Methodology

Figure 3.1: Publication overview concerning a comprehensive system-on-chip evaluation methodology for secure/reliable software and hardware implementations

3.2 Power-Aware and Fault-Aware Software Verification

Software verification under the consideration of its power consumption and robustness against malicious attacks has been individually described in literature, as shown in Section 2.1.1 and Section 2.1.3. Nevertheless, many aspects of security engineering or software verification have only been partially covered in literature. Also, no comprehensive approach using an FPGA-based hardware accelerated solution is available that can be flexibly applied to existing software verification infrastructure.

3.2.1 Data-Aware Emulation-Based Power Estimation

Originally, the power emulation approach, as described in [32], only considered the current control state of carefully selected parts of the system. This power estimation principle is depicted in Figure 3.2, and shows the possibility of direct interpretation and data storage. By principle, such an architecture cannot be directly extended with data-dependent signal, as it would distort the generated power profile. Therefore, a process has to be defined to integrate such information into the existing infrastructure for the on-line and off-line estimation of information leakage. The goal in this case is not the exact determination of such leakage, but the evaluation of embedded systems under different scenarios, such as best and worst case conditions.

The power consumption itself is estimated using macro models defined as described by Equation 3.1. The power estimate $\hat{P}[t]$ consists of a static part $\hat{P}_{sta}$ caused by semiconductor leakage and a dynamic part $\hat{P}_{dyn}[t]$ considering switching activity. While the static power can be estimated by a single coefficient c_0, dynamic shares have to be considered more precisely.

$$\hat{P}[t] = \hat{P}_{sta} + \hat{P}_{dyn}[t] = c_0 + \sum_{i=1}^{Nc} c_i x_i[t] + \sum_{i=1}^{Nd} d_i x_i[t] \tag{3.1}$$

The estimation of dynamic power consumption as suggested by Equation 3.1 is visualized in Figure 3.2. Data signals and control signals are considered by corresponding coefficients d_i and c_i. These are multiplied by the current value of the observed signal x_i. Data signals have to be pre-processed to be considered in the same way as state signals. This pre-processing mechanism is described in more detail later-on.

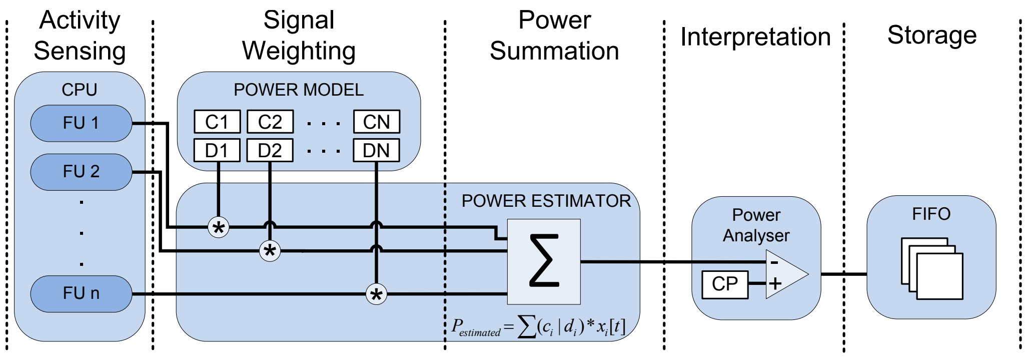


Figure 3.2: Basic power emulation principle - Generating run-time power estimates from the current state of selected control signals (adapted from [5])

Based on existing knowledge concerning the characterization of systems-on-chip using standardized benchmarks, a novel power model creation process has been derived. This methodology is depicted in Figure 3.3, showing that the principle flow and therefore, software support has only been slightly altered. The first step involves the generation of training sets using standard benchmark suites and simulation (functional and RTL power) tools. From this simulation data state and data dependent training sets are separately extracted. These extracted sets are then utilized for *model parameter selection*, followed by a non-negative linear regression based *coefficient fitting* process. The result of this characterization process is a power macro model, including coefficients for state-dependent and data-dependent signals.

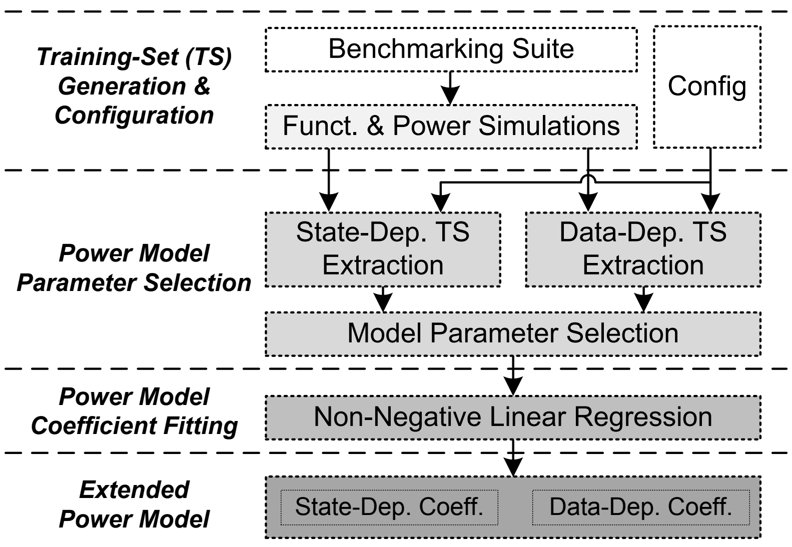

Training-Set (TS) Generation & Configuration
Benchmarking Suite
Config
Funct. & Power Simulations
Power Model Parameter Selection
State-Dep. TS Extraction
Data-Dep. TS Extraction
Model Parameter Selection
Power Model Coefficient Fitting
Non-Negative Linear Regression
Extended Power Model
State-Dep. Coeff.
Data-Dep. Coeff.

Figure 3.3: Power model creation process - Utilization of accurate power simulation to generate reduced power macro models (adapted from [6, 5])

The generated power macro model can now be mapped onto reconfigurable hardware in an automated way, as originally described in [26]. This process has to be adapted to consider the power consuming nature of data signals (CMOS gates only produce power consumption during the switching from one level to another, as depicted in Figure 3.4).

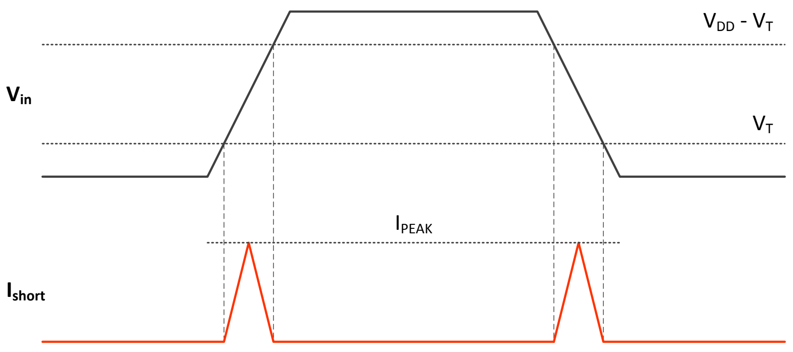


Figure 3.4: Power consumption caused by CMOS gates during signal switching

To map both power consumption models into a single emulation platform, a conversion of the data signals is necessary. This conversion task corresponds with the XOR function of the data-signal with a copy of itself that is delayed by one cycle. The height of the estimated peak current is done by the selection of corresponding power model value. Such an extended power emulation architecture considering these differences is shown in Figure 3.5.

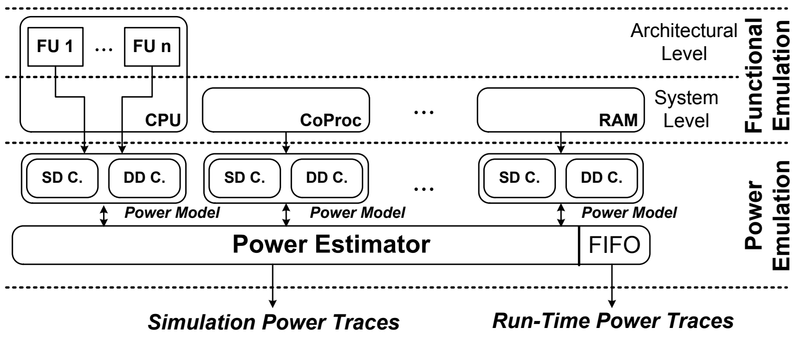


Figure 3.5: Power emulation architecture - Flexible and integratible hardware power estimator (adapted from [5])

These data-aware power estimation characterization and hardware generation concepts are defined more precisely in Sections 6.1 and 6.4.

3.2.2 Software Verification under Fault Conditions

Power consumption and information leakage are only one of a secure software developer's concerns during development. A comprehensive software verification platform has also to provide the possibility of testing the implementation under the influence of faults. For this thesis, an FPGA-based approach has been chosen, to integrate novel investigation techniques with an existing verification platform. This system integration is supported by an automatized process as depicted in Figure 3.6. Before the targeted system can be augmented, attack descriptions and security guidelines have to be studied for the determination of fault attack patterns. The power characterization processes rely on predefined control and data signal sets, which can also be generated using scripts if a specific coding style has been followed. These tasks have to be executed during the *target definition* phase, followed by the *model creation process* utilizing characterization and selection tasks to extract fault and power models. During the *VHDL augmentation* phase these models are integrated into a given system model using pre-defined module templates. Finally, the combined evaluation architecture is transformed into a netlist for the application in an FPGA-based debugging tool using *system synthesis.*

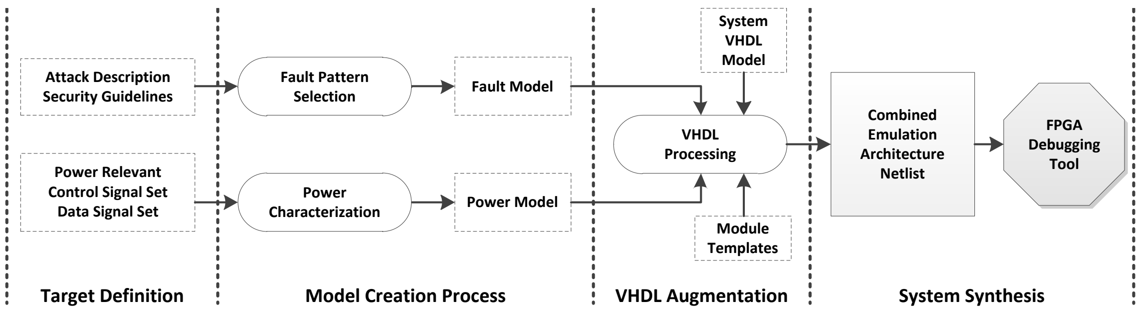


Figure 3.6: Automatized System Augmentation and Verification Platform Generation (adapted from [5])

Depending on the amount of evaluated time slots and integrated saboteurs, a significantly high count of test vectors has to be evaluated. This time consuming process has to be supported by controlling support hardware such as a processor hard-macro to accelerate the investigation approach. Such a high-performance architecture, relying on a fast PowerPC processor, is shown in Figure 3.7. An exhaustive FPGA-based evaluation platform combining all these techniques is described in more detail in Section 6.4.

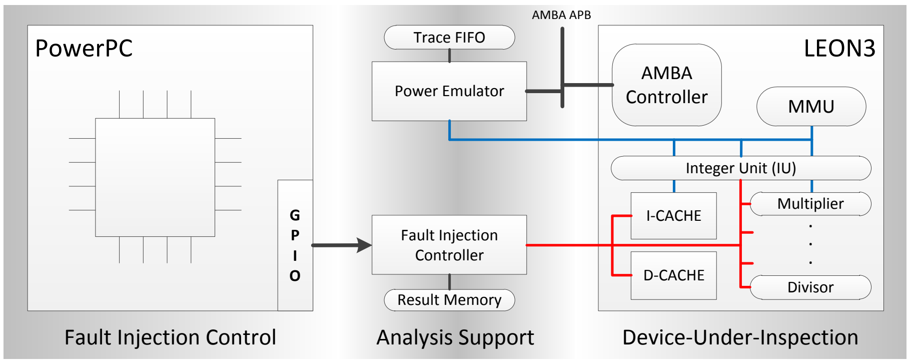


Figure 3.7: High Performance Architecture for the Power Evaluation of Software under Faulty Conditions (adapted from [5])

Up to this point, only very rough side-channel leakage investigations, exploring corner cases like best or worst case scenarios, can be done. More realistic evaluations need information from the physical layout to include information about coupling capacitances, which have an influence on switching power. An extended methodology to integrate physical information, from the *RC extraction* phase of the ASIC creation process, is depicted in Figure 3.8. For the extraction of physical information a completely placed and routed design is needed. The RC extraction task provides various capacitance information from this implementation that can be used to estimate leakage caused by CMOS switching. Such an accurate leakage model is the generated using a given filter configuration and integrated into a leakage sensor module. Finally, these sensors are placed into the RTL design using an automatized process.

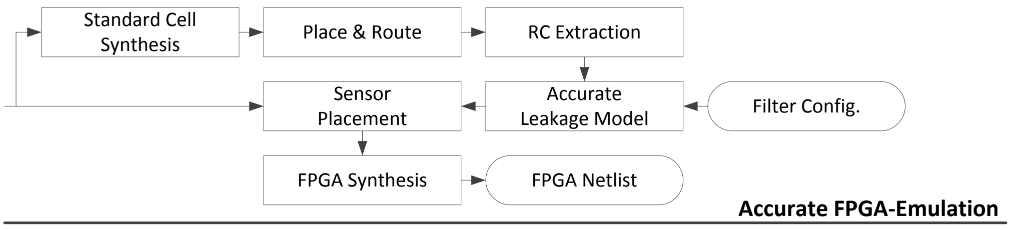


Figure 3.8: Extension of the Data-Aware Power Estimation Approach with Leakage Sensors (adapted from [7])

The presented layout-aware evaluation and hardware preparation flow is presented in more detail in Section 6.5.

3.2.3 Evaluation Support using Static Code Analysis Methods

The vast complexity of large systems-on-chip makes the increasing usage of static code analysis methods inevitable. Also during extensive testing for the application of security and trust certificates, such static methods can be of vital importance to support the mapping of physical effects to the RTL descriptions. A novel methodology integrating static structural analysis of VHDL code for the mapping of results from the laser fault injection testbench is depicted in Figure 3.9. This multi-stage approach starts with the *structural analysis* of the RTL description to extract architectural dependencies between hardware elements. During the next step, physical attack information from a laser testbench is mapped to the hardware constructs defined in the RTL description. This mapping information is used for the automatized generation of emulation support modules. These modules enable the emulation of laser attack effects inside an FPGA based software verification system.

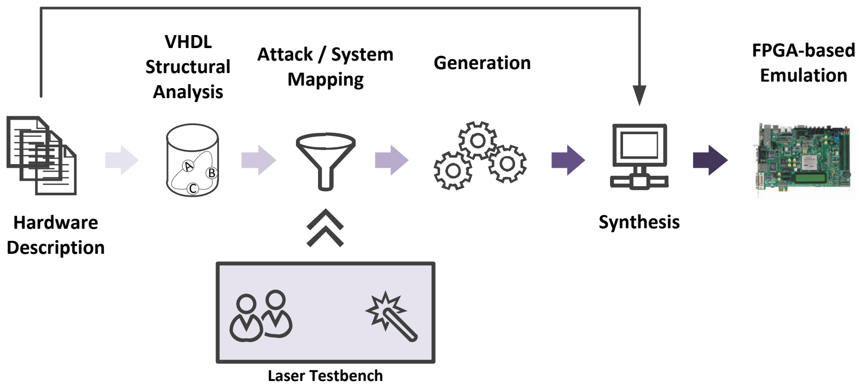


Figure 3.9: Global Fault Attack Analysis Flow under the Consideration of Fault Propagation (adapted from [8])

The shown techniques and methodologies including experimental data are described in Section 6.9.

3.3 Control State Based Fault Detection and Run-time Manipulation

The experiences gathered during the extensive evaluation of processor architectures for their fault attack robustness, is now applied to design of novel system-level fault detection mechanisms. Furthermore, the same infrastructure can be reused to influence information leakage and power consumption during run-time. Under the consideration of existing work in the field of comprehensive techniques utilizing the hardware/software codesign process as summarized in Section 2.2.3, novel signature-based detection mechanisms have been designed.

3.3.1 System-wide Signature-Based Fault Detection

As described in Sections 2.2.1 and 2.2.2, existing approaches either rely on large performance or hardware overhead, or do only cover parts of the checked system. Therefore, in this thesis, a novel methodology is presented to merge on-line power estimation infrastructure with a signature-based fault detection approach to provide low overhead at high detection coverage. Such an characterization and implementation methodology is depicted in Figure 3.10, showing an automated approach for the augmentation of existing system descriptions. This process relies on a pre-selected control signal search space, as it results from former power estimation runs, and contains four processing stages for the automatized generation of hardware augmentations. The first phase consists of two parallel sub-tasks: *power characterization* to determine highly active control regions and *signal activity analysis* for the selection of only relevant signals with deterministic behavior. The result of this stage is a reduced control signal set that can now be utilized for the *signature hardware generation* tasks that also considers certain architectural constraints given by the design engineer. After optimized hardware signature generators have been created, an automatized wiring process connects these to the selected control signals of the target design.

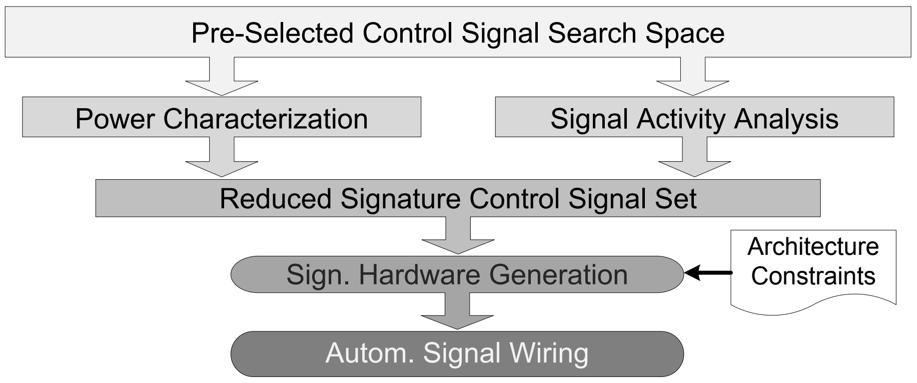

Pre-Selected Control Signal Search Space
Power Characterization
Signal Activity Analysis
Reduced Signature Control Signal Set
Sign. Hardware Generation
Architecture Constraints
Autom. Signal Wiring

Figure 3.10: Control Signal Signature Characterization and Implementation Methodology (adapted from [9])

This automated process results in a combined architecture for the on-line power estimation and checking of pre-generated signatures, as shown in Figure 3.11. Such signatures can be generated during software verification phase, using the same FPGA-based evaluation platform as described in the previous section. The major advantages of this approach are the rather low logic implementation effort and the possibility of saving complete control signal chains. These concepts and methodologies are more precisely presented in Section 6.6.

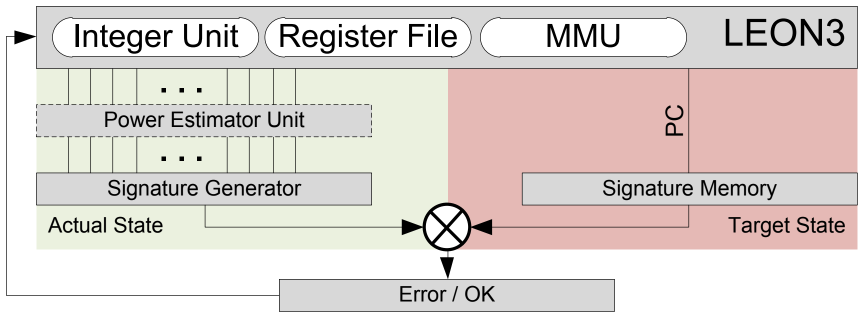


Figure 3.11: Signature Checking Architecture for a General Purpose Processor (adapted from [9])

This architecture still relies on rather large memories for the storage of detailed signature chains. Therefore, a generation process for pure hardware solutions, as depicted in Figure 3.12, is strongly needed. This flow consists of four basic sub-steps: *signature characterization*, *relation extraction*, *boolean minimization* (optional as current synthesis tools fulfill this task quite well), and *automatized routing*. The signature characterization process includes the recording of instruction-control signal signature pairs during a chosen program section. The MATLAB based relation extraction phase determines unambiguous pairs for the further processing. Finally, these pairs are mapped onto a general hardware structure for synthesis. The automated generation flow and implementation details are described in Section 6.7.

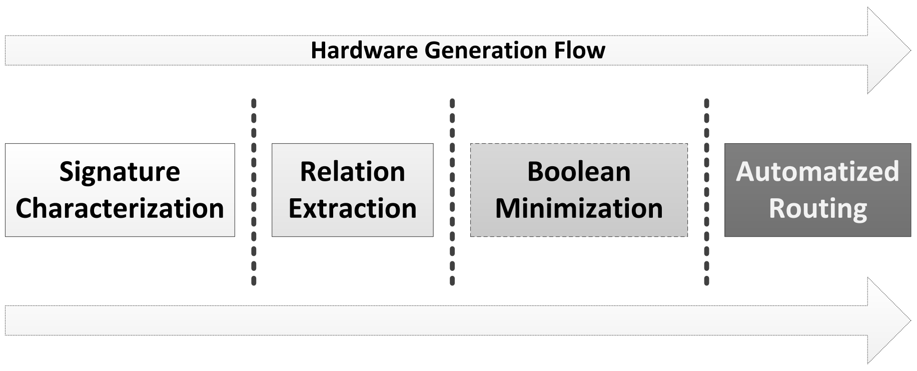


Figure 3.12: Hardware generation process for modular signal monitor architectures (adapted from [10])

3.3.2 Run-time Manipulation of a System's Power Consumption

The existence of a control signal based power estimation and fault detection infrastructure also opens the possibility of on-line methods to directly react to power consumption changes or attacks. Such a power profile scrambling methodology, to decrease the signal-to-noise ratio for the application of power analysis, is shown in Figure 3.13. The power estimation result is averaged and differentiated to retrieve only qualitative statements about the current consumption. For each profile manipulation technique dedicated *look-up-tables* (LUT) are used to determine how strongly these should influence the behavior of the design.

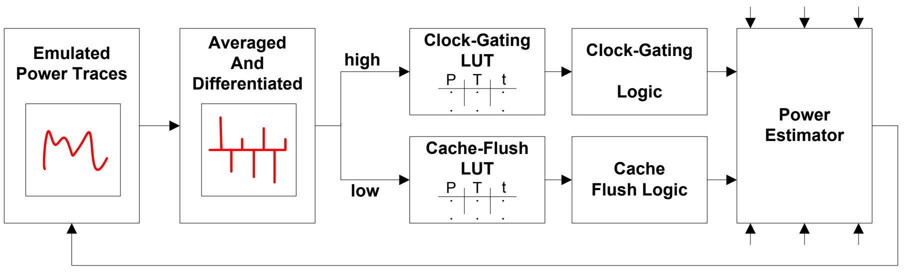


Figure 3.13: Online Power Consumption Manipulation Methodology (adapted from [11])

The presented concepts are described in more detail, including experimental data, in Sections 6.3 and 6.4.

Chapter 4

Evaluation of the MPSoC Fault Detection and Analysis Techniques

4.1 Overview

The techniques and methodologies presented in this work have been evaluated using two different FPGA-based verification platforms. Using these, both, a widely used industrial smart-card platform and an open academic system-on-chip application can be investigated. The results derived during these investigations can be grouped into three major groups: *software verification*, *signature-based fault detection*, and *static analysis methods*. The general applicability of our fault analysis approach is demonstrated by the integration into both FPGA-based platforms. Extensive emulation-based evaluations show the efficiency of the presented fault-attack countermeasure implementations. Finally, static RTL code analysis methods have been applied to support the mapping of laser-attack scenarios to these verification systems.

4.2 Target and Evaluation Platforms

The generality of the techniques presented in this thesis calls for an experimental setup that covers a wide range of target systems-on-chip platforms. First, an industry-grade smart-card implementation based on the SLE77 16-bit security system-on-chip and a corresponding software verification platform have been selected. Second, a general FPGA-based prototyping platform using a Xilinx Virtex5 family FPGA and the 32-bit SPARC v8 based system-on-chip implementation from Aeroflex Gaisler have been chosen.

4.2.1 Software Verification in Smart-Card Systems

This software verification platform is directly built from the RTL description of the SLE77 family smart-card devices. Therefore, it contains the same 16-bit micro-controller that has been specifically designed with power efficiency and fault-attack hardness in mind. Additionally, this system-on-chip contains various peripherals and co-processors, such as symmetric and asymmetric cryptographic cores and interfaces to common buses like I2C or contact-less implementations. This FPGA-based prototyping platform is widely used as a software verification system for internal and external software developers. Hence, it constitutes a perfect example for a typical verification platform with a trust barrier between manufacturer and software designer. This problem of trust is solved by the usage of encrypted net-lists and clearly defined debug interfaces.

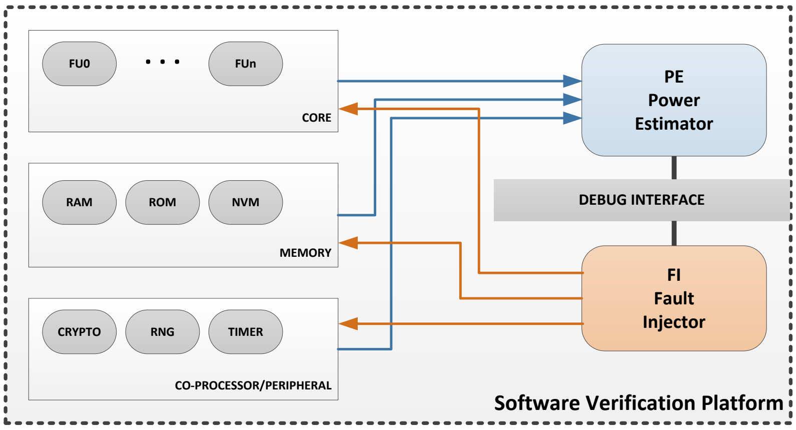


Figure 4.1: SLE77 system-on-chip architecture overview

4.2.2 Multi-Core LEON3 System-on-Chip Platform

The LEON3 processor is a 32-bit SPARC v8 based microarchitecture that is released by Aeroflex Gaisler under an open-source license and therefore, is widely used in academia for the evaluation of novel design and test methodologies. This system-on-chip design has also been proven in various space application and hence, comes with a wide variety of different communication interfaces and platform support modules. For this work, this SoC implementation has been implemented on a Xilinx Virtex5 FPGA including an additional PowerPC processor for process control. The open nature of the design allows for deep changes inside the system design and the hard-macro PowerPC core highly improves execution performance. The processor itself is well supported by open-source software development tools and the Linux kernel. The core is also highly configurable and allows the instantiation of upto 16 processors cores.

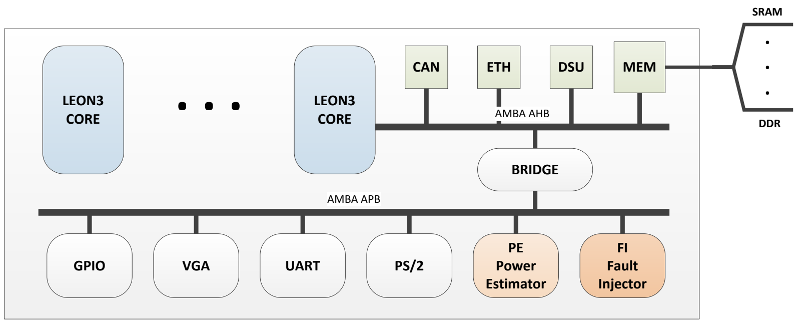


Figure 4.2: LEON3 system-on-chip architecture overview - Obtained with modifications from [12]

4.3 Power-Aware and Fault-Aware Software Verification

The methods introduced in Section 3.2 have been evaluated on both available investigation platforms, to prove general applicability and implementation correctness. This evaluation has been divided into three parts, of which two are security-related and the last one maps the developed methodology to safety investigations.

4.3.1 Hardware-Accelerated Power-Aware Hardware and Software Fault Attack Evaluation

The introduction of power emulation techniques for the run-time generation of power estimates allows for the detection of strong variations in the power profile of secure software implementations. In Figure 4.3 the execution of the dhrystone benchmark on a common smart-card platform using such a software verification approach, is shown.

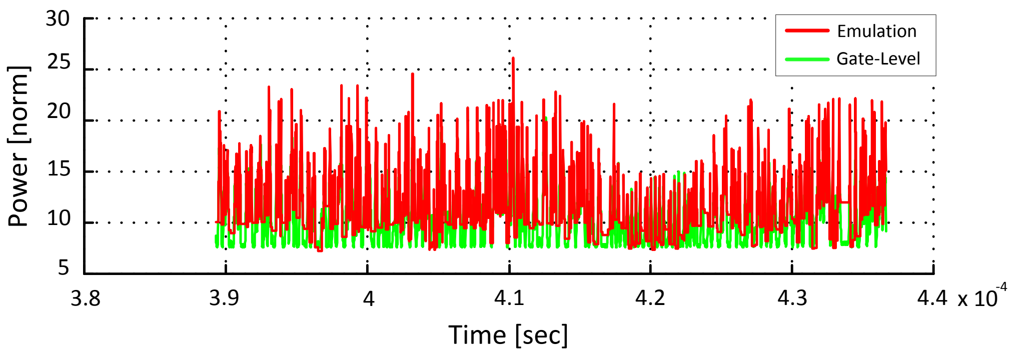
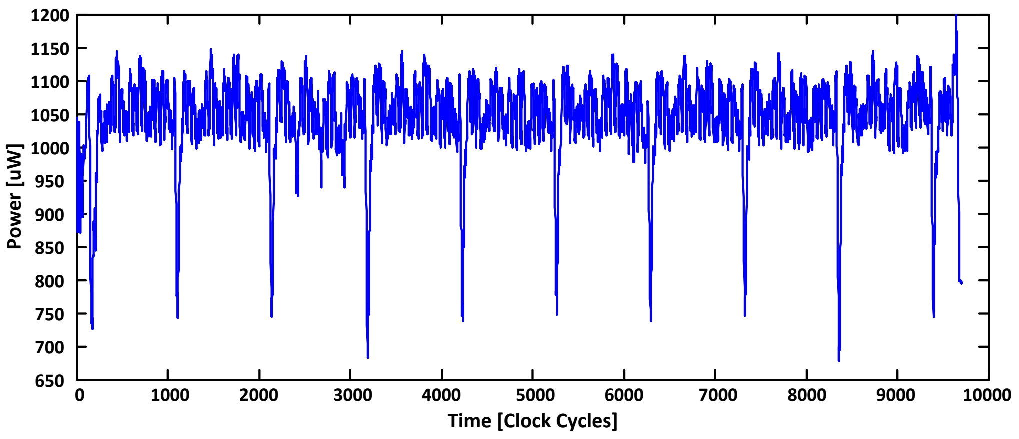
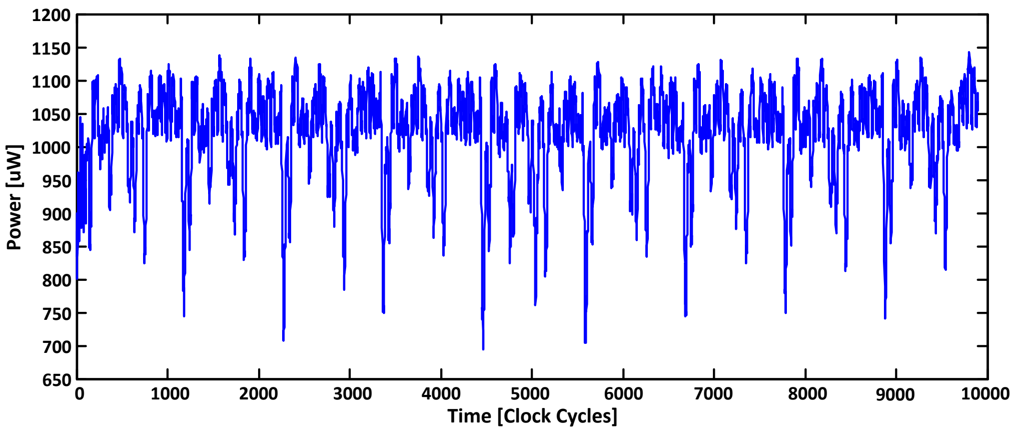


Figure 4.3: Power-aware software verification, run-time power estimation of the dhrystone benchmark program running on the SLE platform - Obtained with modifications from [5]

This control signal based technique has been extended with data-aware information to enable power analysis on the power consumption profile as described in Section 3.2.1. One application of such a mixed evaluation technique is the determination of the effectiveness of novel countermeasures. Applying this technique to a software implementation of the AES algorithm is depicted in Figure 4.4. Sub-figure a shows one encryption run including all 10 rounds without the application of a novel control manipulating countermeasure. The sub-figure b this countermeasure has been activated and it can be clearly seen that the signal-to-noise ratio is significantly increased without any changes to the software implementation. Such a power analysis countermeasure method has been described in detail in Section 3.3.2.



a.) AES software encryption without the application of any countermeasures



b.) AES software encryption under the influence of a clock gating algorithm

Figure 4.4: Power consumption trace of an AES software encryption with and without the application of countermeasures using the LEON3 platform - Obtained with modifications from [5]

As shown in Section 3.2.2 the power-aware software verification approach can now be extended for fault injection investigation by the integration of saboteur modules and corresponding control logic. This technique can also be used for the fault attack hardness evaluation of hardware blocks, such as cryptographic IP blocks.

In Figure 4.5 an attack on the AES algorithm as described in literature, is depicted. This test case shows how a hardware-independent attack can be mapped to any hardware implementation by only placing a few saboteurs at specific points. In this case, a single saboteur is used to disrupt the key scheduling in a defined way.

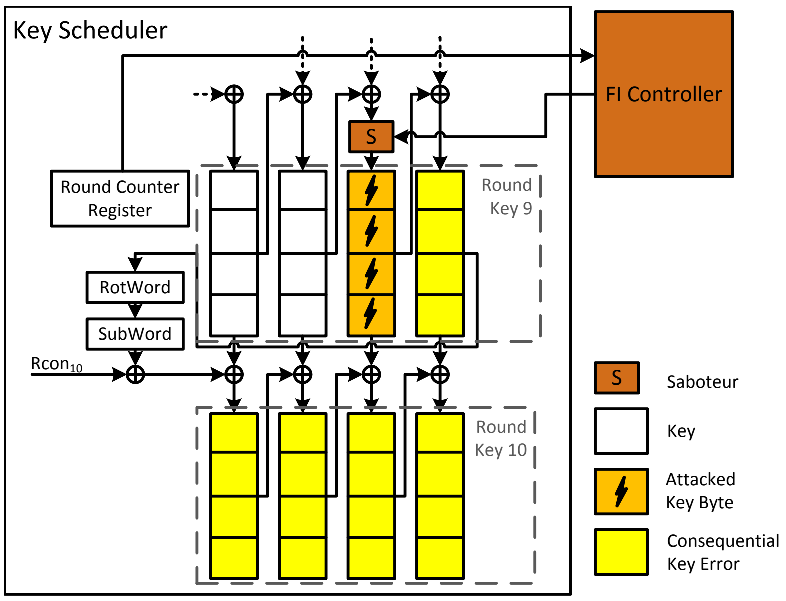


Figure 4.5: Mapping of a fault-attack scenario to a saboteur based fault injection architecture - Obtained with modifications from [5]

The results of such an attack is shown in Figure 4.6. If the key scheduling process is influenced by an attack in the ninth round the original key can be calculated from the corrupted output, as described in [70].

1F	1B	17	13
1E	1A	16	12
1D	19	15	11
1C	18	14	10

(a) Original Key

E0	1B	F7	08
E1	1A	F7	08
E2	19	F7	08
31	84	E9	1C

(b) Corrupted Output

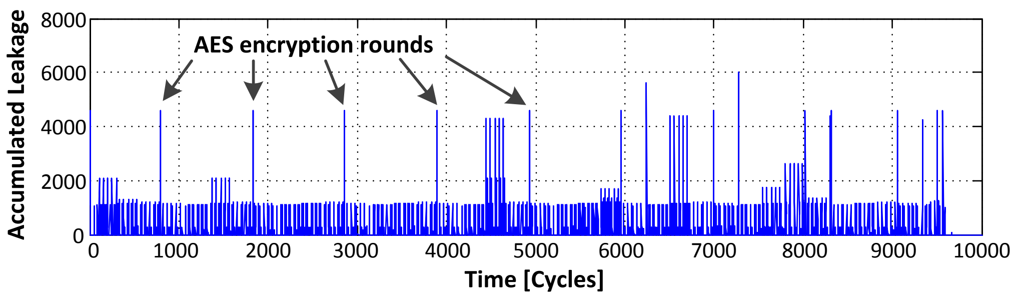
1F	1B	17	13
1E	1A	16	12
1D	19	15	11
1C	18	14	10

(c) Calculated Key

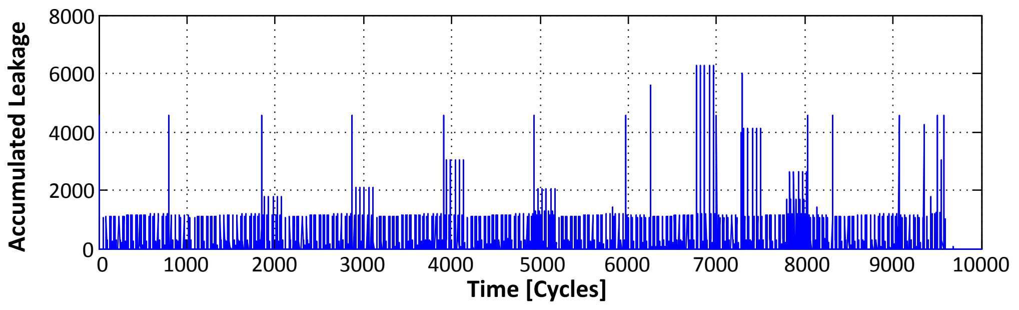
Figure 4.6: AES-SBOX fault emulation results

At this stage information leakage through the power consumption profile could only be emulated in a static way without link to the physical design. Therefore, Section 3.2.2 described a novel strategy to use information from physical capacitance data to generate leakage sensors. The efficiency of such leakage sensors has been investigated for both a software implementation of the AES algorithm as well as a hardware implementation.

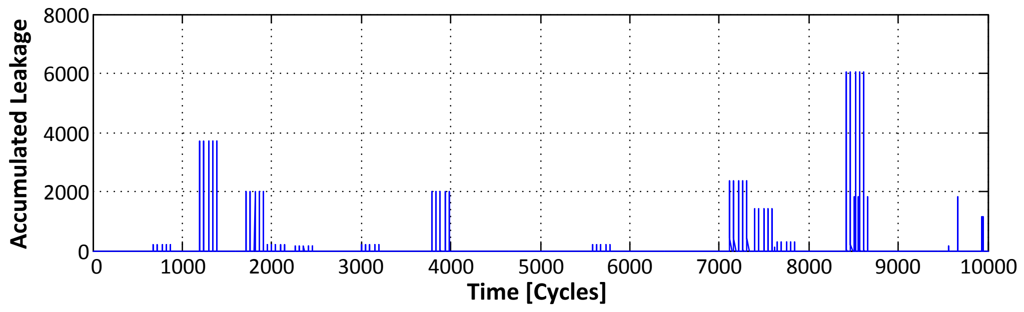
In Figure 4.7 the accumulated sensor results are shown for different randomly chosen plaintexts for encryption and keys for the filtering process as described in Section 3.2.2. Sub-figure a shows an AES encryption using a random plaintext filtered with the correct known key. The information leakage caused by the loading of the key into the processor pipeline can be clearly identified. Sub-figure b just enforces this filtered result. In sub-figure c it can be seen how this filtering approach is depending on a correct key and also shows the applicability of the proposed technique.



a.) AES encryption using random plaintext, filtered with correct key



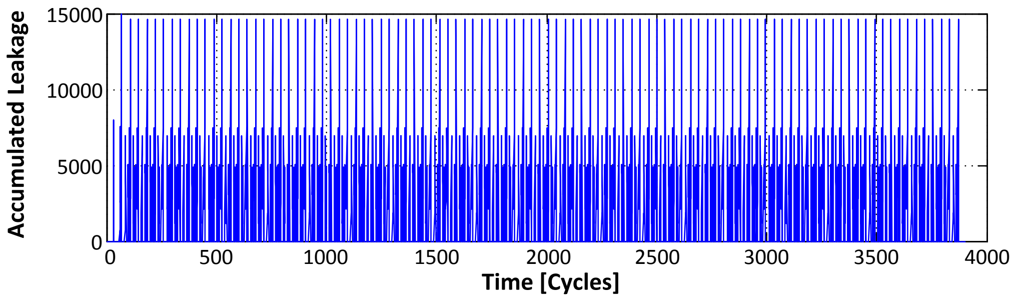
b.) AES encryption using random plaintext, filtered with correct key



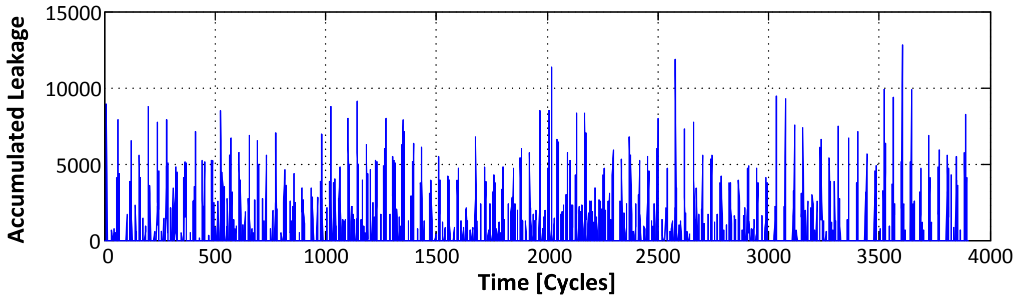
c.) AES encryption using random plaintext, filtered with random key

Figure 4.7: Leakage sensor results for an AES software implementation - Obtained with modifications from [7]

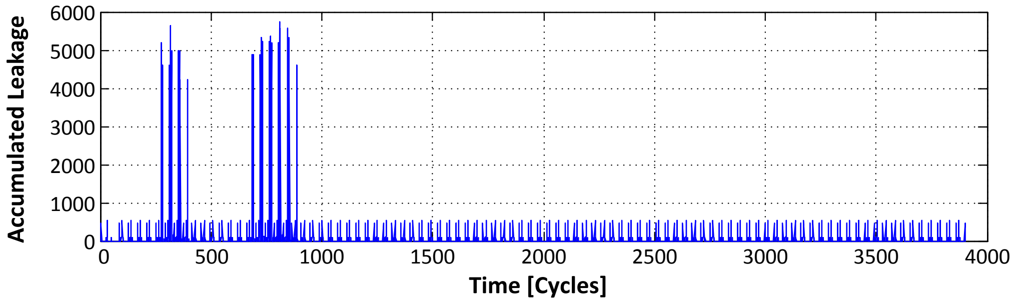
The same approach can also be applied to hardware modules, for example to investigate masking techniques. Such an evaluation is depicted in Figure 4.8. Sub-figure a shows 100 traces recorded from hardware block internal leakage sensors without the application of any masking techniques. This results in visible peaks caused by every AES encryption process. In case of the evaluation run depicted in sub-figure b a simple masking approach has been implemented and this results in a more randomized leakage distribution. Sub-figure c demonstrates why a badly implemented masking scheme isn't protecting a complete system. In this case, the key is leaking through the processor data caches, as in this general purpose implementation the key itself has not been protected.



a.) AES hardware module – 100 traces without masking



b.) AES hardware module – 100 traces with masking



c.) AES hardware module – Processor data cache input leakage

Figure 4.8: Leakage sensor results for an AES hardware implementation - Obtained with modifications from [7]

4.3.2 Emulation-based Secure Software Verification for Smart-Card Products in an Industrial Setting

The evaluation platform presented in the previous section has been tested in an industrial environment to prove its general applicability. This specific case study relies on the emulation of attacks on the memory sub-system of a smart-card. This implementation is depicted in Figure 4.9, mapping given attack scenarios using a saboteur on the cache data bus. The control of these attack runs is provided by a programmable fault injection controller that is connected to the verification system's serial interface. This enables the parametrization through standard APDU (Application Protocol Data Unit) commands.

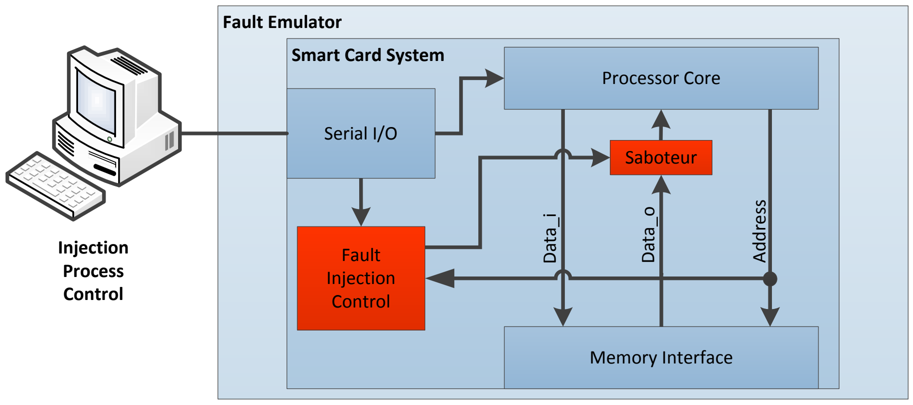


Figure 4.9: Mapping of memory attack scenarios onto an FPGA-based software verification system - Obtained with modifications from [13]

The results of a long-time attack evaluation running for about 22 hours is shown in Table 4.1. Such exhaustive attack campaigns led to various findings: first, over three quarters of the attacked cells are never accessed, second, most of the successful attacks have no effect on generated output values and third, all remaining successful attacks have been successfully detected.

Table 4.1: Long-time attack run using an commercial verification platform

	Number	Percentage [%]
Number of injections [total]	156672	100
Attacked cell never accessed	121570	77.6
Attack resulted into Fault	35102	22.4
No output effect	30373	19.4
Fault detected	4729	3.0

4.3.3 Power-Aware Evaluation of Self-tests in the Safety Domain

The requirements demanded on the software implementation in security and safety domains are often quite similar. Therefore, the same evaluation methodology as described in Section 3.2.2 can be applied to a verification platform for safety critical designs. Such a comprehensive evaluation platform for the testing of software implementations on different implementation levels, is depicted in Figure 4.10.

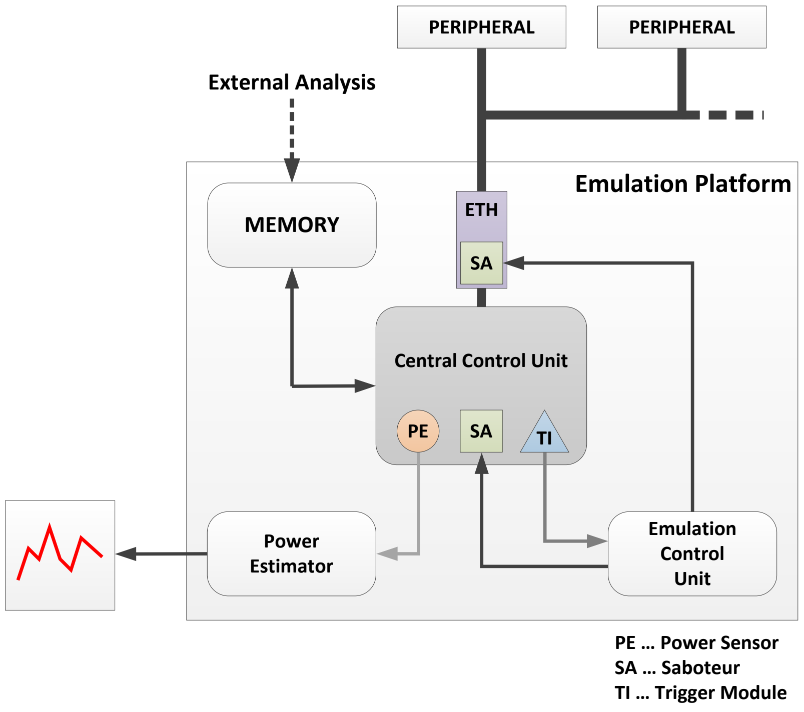


Figure 4.10: Evaluation architecture for the software verification of automotive implementations - Obtained with modifications from [14]

The evaluations itself have been carried out on three different abstraction levels. First, the outside communication of an Ethernet controller has been attacked and the communication packages have been evaluated on the software levels. These results are concluded in Table 4.2 for two benchmarks of the MiBench Benchmark suite. In this case, damaged packages are detected by the IP hardware block or the client Ethernet card in every case, resulting in lost packages, which can be detected by the usage of sequence numbers.

Table 4.2: Fault injection into Ethernet communication results [14]

Benchmark	Injected Faults	CRC Fails	Sent Packages	Received Packages	Lost [%]
Basicmath_small (Client)	>255000	0	13860	9761	29.6
Basicmath_small (Server)	>255000	0	9761	9761	-
Basicmath_large (Client)	>162000	0	1900	302	84.1
Basicmath_large (Server)	>162000	0	302	302	-

During the next attack campaign faults inside the Ethernet controller's FIFO memories have been emulated. As the results in Table 4.3 show, these defects are detected by the CRC mechanism of the Ethernet protocol. Furthermore, damaged packages are detected by the controllers, again resulting in lost packets.

Table 4.3: Fault injection into Ethernet controller memory results [14]

Benchmark	Injected Faults	CRC Fails	Sent Packages	Received Packages	Lost [%]
Basicmath_small (Client)	> 260000	10	7929	7019	11.5
Basicmath_small (Server)	> 260000	835	7094	7929	-
Basicmath_large (Client)	> 248000	3	1608	1361	15.4
Basicmath_large (Server)	> 248000	157	1451	1608	-

The same benchmark applications have finally been attacked by fault injections into embedded processor memories. As seen in Table 4.4 such malfunctions cannot be covered by simple protocol error checking and hence, result in corrupted outcomes.

Table 4.4: Fault injection into embedded processor memory results [14]

Benchmark	Injected Faults	Calculations	Corrupted	Corrupted [%]
Basicmath_small (Client)	> 1460000	5637182	590432	10.47
Basicmath_large (Client)	> 791000	3048202	264451	8.68

In modern safety systems, processor defects are detected using self-test procedures covering a wide range of processor elements. Such self-tests have been investigated and the results are shown in Table 4.5. It can be seen that almost in all cases, all injected faults have been correctly detected.

Table 4.5: Evaluation of embedded processor self-test procedures [14]

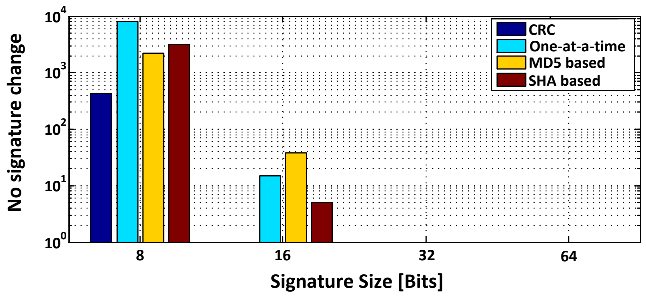
Self-test	Injection Type	Faults Injected	Detected [λ_S]	Fault Coverage [SFF]
Multiplier	Stuck-At 1	1000	1000	100%
Multiplier	Transient 1	1000	1000	100%
ALU	Stuck-At	1000	1000	100%
ALU	Transient	1000	1000	100%
Shifter	Stuck-At	1000	1000	100%
Shifter	Transient	1000	999	99.9%

4.4 Signature-based Fault Detection in MPSoCs

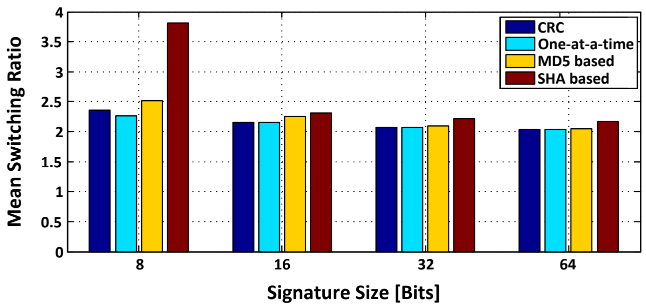
In Section 3.3 novel methodologies for the generation and implementation of area efficient signature generators are described. These techniques have been thoroughly evaluated using the LEON3-based system-on-chip platform introduced in Section 4.2.2.

4.4.1 Characterization of Systems for Optimized Signature Implementations

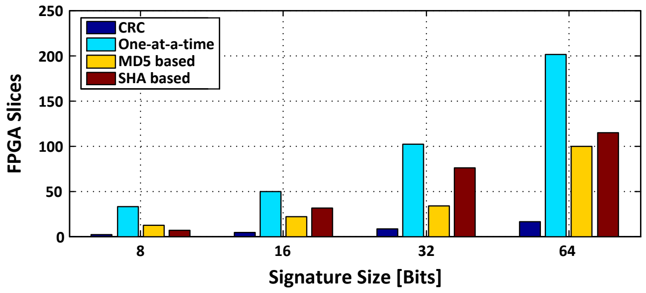
The basic idea behind the signature approach introduced in this work, is the breaking up of existing hash algorithms. This process has been chosen to retrieve hash generators that fulfill basic properties of well defined signature algorithms at highly reduced hardware cost and low generation latencies. These hash properties have to be re-determined using a characterization process, which results for different implementations are depicted in Figure 4.11.



a.) Evaluation of hash algorithm implementations for signature collisions



b.) Evaluation of hash algorithm implementations for their switching ratio



c.) Evaluation of hash algorithm implementations for their hardware effort

Figure 4.11: Evaluation of various reduced hash algorithm implementations, depending on different signature sizes - Obtained with modifications from [9]

During this investigation process, four different signature hardware implementations have been evaluated for their susceptibility to collisions (different input vectors share the same generated signature), the input-output bit switching ratio and the needed hardware implementation effort after FPGA synthesis. These tests have been executed for four different signature widths to identify eventual corner cases for very small and very large signatures. In Sub-figure a it can be seen that a CRC-based implementation has the most favorable properties with very low collisions for the smallest signature sizes. Sub-figure b depicts the measure input-output bit switching ratio, resulting in very similar results for all implementations with exceptionally good results for the one-at-a-time design for the smallest hash class. As seen in Sub-figure c in terms of hardware implementation effort the CRC-based generators require the lowest amount of logic.

To compare our proposed hardware architecture with existing designs described in literature, we integrated our generators into a LEON3 system. As shown in Table 4.6 both derived hardware blocks, the generator and controller, result in insignificant hardware effort. This implementation is based on pre-computed control signal hashes and a run-time checking logic.

Table 4.6: Resource usage comparison [9]

Unit	Slices	Slice Regs	LUTs	Resource [%]
System	15937	15017	30768	-
Core	3342	3248	6633	21.6
Signature Generator	103	102	263	0.85
Signature Controller	234	146	451	1.47
Arora et.al [71]	-	-	-	3-9[a]

[a]In [71] several different techniques have been proposed

4.4.2 Automatized Generation of Signature Checking Hardware

A large portion of on-line checking implementations described in literature are based on pre-computed signatures or control flow trees to detect execution disruptions. In resource-constrained system the application of such designs using rather large memories could be undesirable. Therefore, in Section 3.3.1 a pure logic-based approach is described to generate hardware blocks predicting expected signatures for given input vectors. The integration of such a monitoring module into a processor's pipeline is depicted in Figure 4.12.

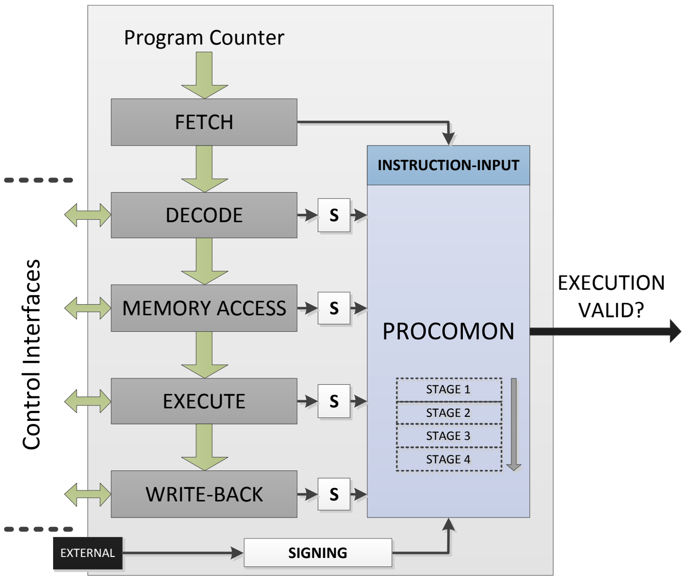


Figure 4.12: Signature-based on-line monitoring architecture for a processor pipeline - Obtained with modifications from [10]

While this implementation example only describes a processor pipeline monitor, it is also applicable to a full system monitoring approach. Such an integration is depicted in Figure 4.13 constituting a simple data accumulation and emergency reset design.

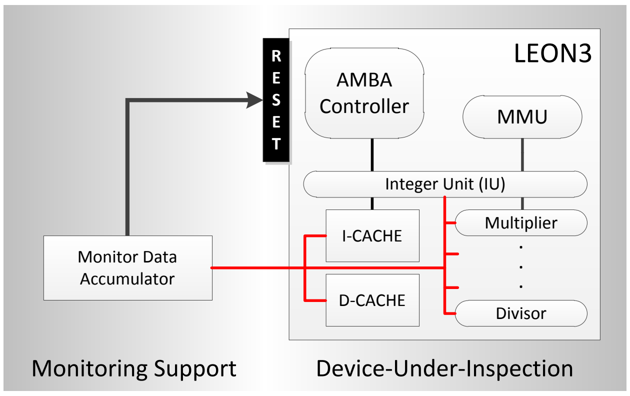


Figure 4.13: Full system checking architecture - Example implementation for the LEON3 system-on-chip - Obtained with modifications from [10]

This monitoring block is generated by the utilization of a software characterization process. During this task, unambiguous pairs of input instructions and resulting control signal signature are searched. The result of such a characterization for different benchmark applications is shown in Table 4.7. The difference between these applications comes from their diverging execution behavior.

Table 4.7: Characterization results [10] - Instruction/Signature Pairs

Application	Reg-Pairs[b]	Exe-Pairs[b]	Mem-Pairs[b]	Xce-Pairs[b]
CoreMark	287	276	83	74
Dhrystone	100	142	80	64
basicmath[a]	145	344	141	141
bitcount[a]	87	64	92	98

[a]Round-reduced part of MiBench embedded systems benchmark suite
[b]Reg(register access), Exe(execution), Mem(Memory access), Xce(exception) stage

The former introduced processor pipeline monitoring approach has been implemented for our LEON3-based FPGA platform and has been compared to existing techniques described in literature. These results are summarized in Table 4.8, showing that this novel design requires a comparable hardware effort compared to existing implementations.

Table 4.8: Resource usage comparison on the LEON3 platform [10]

Unit	Slices	Slice Regs	LUTs	Resource [%]
System	7613	7413	15378	-
Core	3903	3567	7841	50.99
Signature Generators	28	109	102	0.66
PROCOMON[b]	407	0	1114	7.24
Arora et.al [71]	-	-	-	3-9[a]

[a]In [71] several different techniques have been proposed
[b]Depending on amount of signature-instruction pairs (bit_count)

Also the detection efficiency and latencies have been determined and compared to existing work. As seen in Table 4.9 both performance targets have been met.

Table 4.9: Fault detection efficiency comparison [10]

Approach	Det. Bit Flips [%]	Det. latency [Instr.]
This work	> 99	1
Mao et.al Control flow [49]	26	23.6
Mao et.al Hash4 [49]	94	1
Arora et.al [71]	>99	6

4.5 Hardware Testing Support using Static Analysis Methods

Laser-based certification testing and the increasing complexity of novel designs made the continuing introduction of formal methods into the verification process inevitable. For this task a Java-based parser and analyzer for RTL designs written in VHDL has been developed. As introduced in Section 3.2.3, these tools allow for the determination of static dependencies inside the design and hence, can be used to extract and analyze fault propagation paths. An example for the analysis of the activation and propagation paths of an attack on the physical design is depicted in Figure 4.14. In this case the physical location of the attack has been translated to an attacked register (**r**) on RTL-level.

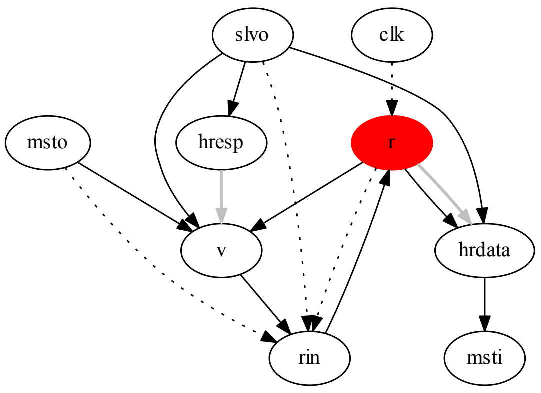


Figure 4.14: Dependencies for an attacked circuit node - Obtained with modifications from [8]

This information can now be used for the automated generation of hardware devices mapping the system-internal effects onto an emulation-based verification platform. As visible in Figure 4.15 the results of such an investigation could be the determination of fault-sensible time regions of a given software implementation. This data can be used to significantly reduce fault attack investigation times or to harden the software implementation against attacks.

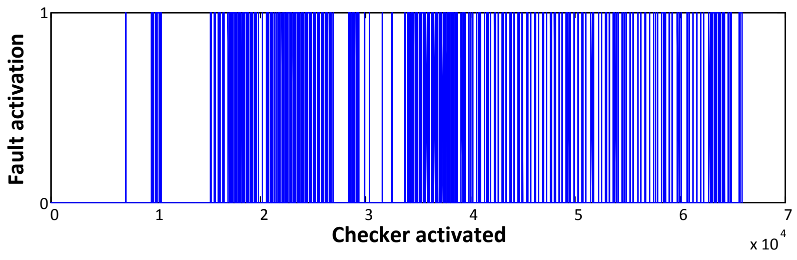


Figure 4.15: Checker results for a specific attack point - Obtained with modifications from [8]

The importance of such tests can be especially seen in Table 4.10 showing that in only 11% of all checker activations, the attack target was sensible to the attack.

Table 4.10: Checker evaluation result for a single attacked node [8]

Checker Name	Checker activ.	Fault sens.	Fault sens. [%]
AHBCTRL_HRDATAS	65896	7261	11

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Highly integrated multi-processor SoCs have been increasingly introduced into every day's life in recent years. These systems cover both ends of the performance scale, from high-performance implementations integrating various additional co-processors and peripherals to low-performance smart-card systems with strict security constraints. Both design targets, high performance or security and reliability through the extensive use of parallelism, introduce a wide range of novel challenges concerning power consumption and executions security. Furthermore, besides theses hardware related challenges such complex systems also come with highly optimized software implementations to implement the various applications running on these platforms. Therefore, the hardware/software co-design process plays a crucial role in managing all these performance and integrity tasks under the watchful eye of certification bodies, in case of high-security or safety applications.

This work engages this hardware/software co-design process on different levels to enable power-efficient and secure/safe products from design to early evaluation and verification. First, a novel software verification platform based on hardware accelerated emulation techniques is described. This technique enables the early investigation of power consumption and fault-attack hardness. By this integration of both evaluation techniques into a single FPGA-based verification platform, both hardware and software designers are enabled to ensure the compliance of the derived implementation to given power and security constraints. The power estimation infrastructure is based on state-of-the-art control-state-based power estimation techniques. Furthermore, it is enriched with data-aware mechanisms to emulate information leakage through the power consumption profile at extreme border line conditions. While the amount of information leakage is defined by the verifying engineer, methodologies to include data from the physical implementation process are also covered by this work.

The heterogeneous nature of complex multiprocessor SoCs and cost constraints in high security smart-card implementations call for novel fault detection techniques. Based on our novel verification platform, a signature-based fault-detection mechanism has been developed to allow for the efficient integration of multiple signature-generators into existing designs. Each signature generator is fed with control information from selected control signals inside modules-under-test. This results in representative signatures every clock cycle that can be pre-recorded to generate a reference stream, or they are generated at run-time to be used in tightly connected redundant architectures. This approach provides much higher coverage of complex systems and faster reaction times than existing techniques which are only comparing module outputs. The integration into the design is supported by automatized routing processes and optimum implementation efficiency is provided by a characterization mechanism. The increasing complexity of novel MPSoC designs and the continuously stricter certification process make the introduction of static system analysis methodologies inevitable. This development is covered in this thesis by source code analysis tools for the VHDL description language and automatized processes for the evaluation of physical implementation data. These techniques allow for both, the direct mapping of fault attacks from physical laser injections to emulation platforms and the increased application of formal techniques for the design verification.

5.2 Directions for Future Work

5.2.1 High-Level Evaluation of Security Policies during Early Design Phases

Trust between manufacturer and customer of high security products is a major challenge during and after the design of novel implementations. System certification aims at providing a defined level of trust between all participants of the supply chain of security critical products. Such assurance levels are defined by Common Criteria to describe how well documented the design process (besides other processes) of such a product is. In case of smart-cards, high assurance levels make it necessary that certain fault attack scenarios are tested by independent test laboratories. This work aims at providing a software verification platform to prepare software and hardware implementations for such tests and real-world attacks. Unfortunately, such emulation-based testing as well as physical tests cannot guarantee that the investigated system is covering all possible attack scenarios. Furthermore, the complexity of smart-card systems is also steadily increasing and hence, the evaluation space is increasing. This continuously increasing verification problem creates a strong need for an extended use of formal methods during the design and verification phases of the implementation. Recent years showed an increased effort in academia and industry to solve these problems, but no comprehensive solutions responding to the specific needs of this industry sector have been found.

5.2.2 System-Level Power and Security Evaluations of Mobile Systems

The massive introduction of smart-cards into the payment and personal ID sectors came with an increasing introduction of mobile reader systems to communicate with these devices. While power and security related challenges in smart-card and reader systems are well known, research concentrated on the resolution of only these isolated problem domains. The interaction between a contact-less smart-card and the reader system and challenges resulting from this wireless connection have not been well investigated. This results into possible security problems and power consumption issues on reader as the transmission power is kept at a maximum level. This results from the fact that currently neither smart-card nor reader system have information about the transmission channel during run-time to react accordingly.

The META[:SEC:][1] collaborative research project [72] aims at solving this research gap between low-level circuit optimizations on smart-card side and power-constraints of portable reader systems. This effort is supported by the long-time experience of the project partners in their specific fields. The continuing research collaboration between industry and university also fosters an increasing aggregation of knowledge concerning the handling of power- and security constraints in reader/smart-card systems.

[1]*Mobile Energy-efficient Trustworthy Authentication Systems with Elliptic Curve based SECurity*, collaborative research project of the Graz University of Technology, Infineon Austria AG and Enso Detego GmbH. Funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract FFG 829586.

Chapter 6

Publications

This chapter presents a brief collection of publications containing a more detailed description of the presented methodology shown in Chapter 3. They also include a very comprehensive overview of the current state-of-the-art as introduced in Chapter 2 and introduction in several sub-topics as presented in Chapter 1. Various experiments have been conducted to show the evaluations introduced in Chapter 4 in a more explicit way.

Publication 1: Krieg et al., *Accelerating Early Design Phase Differential Power Analysis Using Power Emulation Techniques*, 4th IEEE International Symposium on Hardware-Oriented Security and Trust 2011 (HOST '11), San Diego, USA, June, 5th – 6th 2011.
Publication 2: Grinschgl et al., *Case study on multiple fault dependability and security evaluations*, Elsevier Microprocessors and Microsystems (MICPRO 1976), accepted for publication, 2012.
Publication 3: Krieg et al., *A Side Channel Attack Countermeasure using System-On-Chip Power Profile Scrambling*, 17th IEEE International On-Line Testing Symposium 2011 (IOLTS '11), Athens, Greece, July, 13th – 15th 2011.
Publication 4: Krieg et al., *POWER-MODES - POWer EmulatoR and MOdel based Dependability and Security evaluations*, ACM Transactions on Reconfigurable Technology and Systems (TRETS), Volume 5, Issue 4, Article 19, 2012.
Publication 5: Krieg et al., *System Side-Channel Leakage Emulation for HW/SW Security Coverification of MPSoCs*, 15th IEEE Symposium on Design & Diagnostics of Circuits & Systems 2012 (DDECS '12), Tallinn, Estonia, April, 18th – 20th 2012.
Publication 6: Krieg et al., *Characterization and Handling of Low-Cost Micro-Architectural Signatures in MPSoCs*, 17th IEEE European Test Symposium 2012 (ETS '12), Annecy, France, May, 28th – June, 1st 2012.
Publication 7: Krieg et al., *PROCOMON - An Automatically Generated Predictive Control Signal Monitor*, 15th Euromicro Conference on Digital System Design 2012 (DSD '12), Izmir, Turkey, September, 5th – September, 8th 2012.
Publication 8: Krieg et al., *Power And Fault Emulation For Software Verification and System Stability Testing in Safety Critical Environments*, IEEE Transactions on Industrial Informatics (TII), Volume 9, Issue 2, 2013.
Publication 9: Krieg et al., *Acceleration of Fault Attack Emulation by Consideration of Fault Propagation*, 11th International Conference on Field-Programmable Technology 2012 (FPT '12), Seoul, Korea, December, 10th – December, 12th 2012.

The techniques described in Chapter 3 and the results presented in Chapter 4 have been disseminated in various publications. The fields investigated during the realization of this thesis can be grouped into the following sectors: *FPGA-based Fault Emulation*, *On-line Detection, Analysis and Manipulation*, and *Off-line Implementation Analysis*. The selected publications have been assigned to several sub-topics as depicted in Figure 6.1.

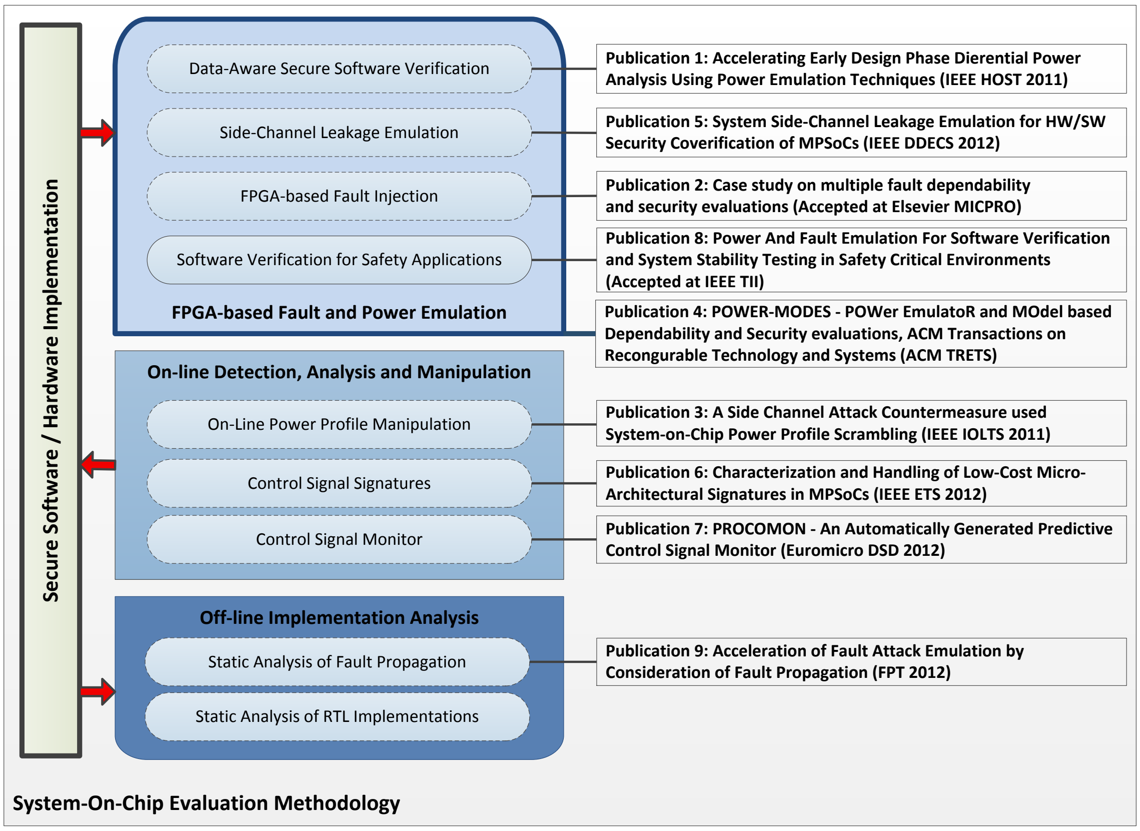

Secure Software / Hardware Implementation
Data-Aware Secure Software Verification
Side-Channel Leakage Emulation
FPGA-based Fault Injection
Software Verification for Safety Applications
FPGA-based Fault and Power Emulation
On-line Detection, Analysis and Manipulation
On-Line Power Profile Manipulation
Control Signal Signatures
Control Signal Monitor
Off-line Implementation Analysis
Static Analysis of Fault Propagation
Static Analysis of RTL Implementations
Publication 1: Accelerating Early Design Phase Dierential Power Analysis Using Power Emulation Techniques (IEEE HOST 2011)
Publication 5: System Side-Channel Leakage Emulation for HW/SW Security Coverification of MPSoCs (IEEE DDECS 2012)
Publication 2: Case study on multiple fault dependability and security evaluations (Accepted at Elsevier MICPRO)
Publication 8: Power And Fault Emulation For Software Verification and System Stability Testing in Safety Critical Environments (Accepted at IEEE TII)
Publication 4: POWER-MODES - POWer EmulatoR and MOdel based Dependability and Security evaluations, ACM Transactions on Recongurable Technology and Systems (ACM TRETS)
Publication 3: A Side Channel Attack Countermeasure used System-on-Chip Power Profile Scrambling (IEEE IOLTS 2011)
Publication 6: Characterization and Handling of Low-Cost Micro-Architectural Signatures in MPSoCs (IEEE ETS 2012)
Publication 7: PROCOMON - An Automatically Generated Predictive Control Signal Monitor (Euromicro DSD 2012)
Publication 9: Acceleration of Fault Attack Emulation by Consideration of Fault Propagation (FPT 2012)
System-On-Chip Evaluation Methodology

Figure 6.1: Publication overview concerning a comprehensive system-on-chip evaluation methodology for secure software and hardware implementations

The first publication shown in Section 6.1 extends the work of the POWERHOUSE project to allow data-aware investigations of secure software implementations. In Section 6.2 a modular fault-injection system is presented and a case-study given by our industrial partners. This work illustrates the applicability of our approach in an industrial setting and the lessons learned at this stage will be used for all following implementations. The possibility of direct system-internal power estimation has been exploited to manipulate a system's consumption in order to close this side-channel. This approach, using only internal resources for manipulation, is presented in Section 6.3 in more detail. The applicability of these techniques, fault-injection, power emulation, and manipulation have been combined to enable comprehensive security evaluations in Section 6.4. Complex system-on-chips contain a wide variety of interfaces that could suffer from possible information leakage, therefore, a modular monitoring and evaluation approach is being reported in Section 6.5.

The combined work of all previous sections created a powerful emulation platform for power consumption investigation under fault influences. Such a platform allows for the evaluation of novel fault detection approaches like the signature-based one shown in Section 6.6. This methodology utilizes control-signal monitoring infrastructure to provide efficient integrity checking hardware. Furthermore, an automatized hardware-generation technique for such checker modules is presented in Section 6.7. Power consumption and fault-effect robustness are not only a concern in security-related products, but also in implementations in the safety domain. To conquer such challenges, several standards have been introduced (such as the IEC 61508 and ISO 26262). The application of the presented fault-emulation and power-emulation platform for the investigation of such safety critical software implementations is being described in Section 6.8. Finally, the application of static code analysis, integrating information from the physical design flow, is introduced in Section 6.9.

Accelerating Early Design Phase Differential Power Analysis Using Power Emulation Techniques

Armin Krieg, Christian Bachmann, Johannes Grinschgl, Christian Steger and Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology, Austria
{armin.krieg, christian.bachmann, johannes.grinschgl, steger, rweiss}@tugraz.at

Josef Haid
Design Center Graz
Infineon Technologies Austria AG
Graz, Austria
josef.haid@infineon.com

***Abstract*—The personal banking and ID sector has seen a tremendous change in recent years, partially caused by the widespread introduction of smart-cards. Because of the extensive implications of a successful attack on these devices, a wide range of practical as well as purely academic attacks has been developed during the last years. These attacks have unveiled weaknesses in hardware as well as software implementations of several different, partially widely used cryptographic algorithms. An especially powerful method, the differential power analysis (DPA), extracts secret information from power consumption and electro-magnetic emission profiles. The efficiency of a DPA attack significantly depends on the quality of the cryptographic algorithm implementation. These traces currently can only be generated using real hardware or simulation-based approaches. Depending on the chosen simulation accuracy these evaluations result in time-consuming RTL and SPICE simulations often limiting the maximum amount of available execution traces. This paper introduces a novel high-speed methodology for early security evaluations of integrated processor systems using power emulation. First, the usage of power emulation hardware allows for the estimation of attack effort that an adversary will have to invest to gain secret information from an algorithm's execution profile. Second, countermeasures against differential power analysis attacks can be quickly evaluated in terms of effectiveness. The shown approach uses semi-automatic characterization techniques and fully synthesizable emulation hardware to reduce the designer's dependency on time-consuming simulation runs.**

I. Introduction

The common availability of power analysis attacks poses a difficult challenge to system designers. Realistic power consumption profiles of a given system-under-test for complex test cases are only available at late design stages, i.e., after first tape-out silicon is available and power measurements can be performed. However, to prevent information leakage in the first place, the system needs to be designed with possible information leakage in mind. The same applies to embedded software developers involved in a hardware/software co-design flow who have to guarantee that their implementation does not compromise a secured hardware architecture. To close this information gap in the system design process several simulation approaches have been proposed. These are based on the assumption that information leakage is caused by two major contributors:

- The root of power leakage is the switching behavior of the used logic style [1]
- Instructions cause different power consumption profiles depending on their complexity [2]

Designers of cryptographic hardware implementations are mostly concerned about leakage caused by a given logic implementation because many cryptographic algorithms are implemented directly in hardware due to performance requirements. On the logic-level evaluations are very time-consuming as they are based on complex, low-level simulation models. Software designers mostly rely on higher level instruction set simulators to verify functionality and, in some cases, to also extract the power profile of a given algorithm. While these high-level approaches offer the benefit of faster simulation times, allowing for the simulation of complex test-cases, they entail the risk of concealing low-level hardware effects and, hence, information leakage.

Especially advanced DPA attacks have become a serious problem in a secure hardware/software co-design process [3]. These attacks rely on a very large number of power traces recorded at very high resolutions. Using low-level simulations it is difficult to provide the required amounts of power traces within reasonable simulation times. Instruction-level simulators, however, do not provide bit- and cycle-accurate processing information, hence limiting their use in DPA-resilience evaluation.

In this paper, we therefore propose the use of state-of-the-art power emulation hardware to provide hardware-accelerated run-time power tracing capability at reasonable levels of resolution. An automated characterization flow facilitates the integration of the power emulation methodology into the design process and allows for the adaption of the approach to novel hardware architectures. Our proposed solution provides additional tools to support the side channel evaluation on different abstraction levels as proposed in [3]. The main contributions of this work are:

- A semi-automatic switching activity-aware power characterization process.
- The introduction of a data-dependent power model for

hardware accelerated security evaluations.
- A case study using a commonly used software implementation of the AES algorithm.

This paper is structured as follows. Section II briefly reviews the current state-of-the-art concerning work on simulation and emulation techniques to evaluate cryptographic software and hardware implementations. Section III gives an overview over power analysis approaches to retrieve secret information from any given system by analyzing its power consumption profile. Section IV provides technical details about the used power emulation approach and necessary changes to enable DPA using such a emulation technique. In Section V the proposed DPA analysis flow is being described in detail. Section VI illustrates experimental results using RTL simulations and the FPGA-based emulation platform. Finally, some concluding remarks concerning the results of this publication are given in Section VII.

II. Related Work

In terms of related work three distinct methods for analyzing the characteristics of a system regarding power analysis characteristics have been presented. They all represent simulation-based approaches.

The first approach comprises a purely software-based solution [4]. For extracting side channel leakage information from a given program, it is executed and every instruction and processor state is traced. This information is fed into an easily extensible analysis module. Such analysis modules are directly implemented for the Microsoft Debugger API. The authors do not state any information about accuracy and simulation performance.

The second approach uses in-system knowledge provided by power consumption simulators using instruction set simulators. While promising high simulation speeds this approach is mostly limited to thoroughly profiled general purpose architectures. Implementations of this approach were presented in [5] and [6]. Both solutions promise high simulation speeds, but do not present any hints about the accuracy of the used power profiles. Inaccurate power profiles could lead a software engineer into a sense of false security as the possibility of missed information leakage could be higher than anticipated. The implementation presented in [6] uses several abstract power models to isolate possible sources of leakage. The presented approach has been designed for the Atmel AVR microcontroller series.

The third approach uses highly accurate gate- or even transistor-level simulations to identify critical parts of the circuit-under-test. This level of accuracy comes with high costs in terms of simulation speed and needed simulation equipment. The designer therefore has to choose between very short simulation times or reduction of the evaluation to the smallest possible leakage contributer. The former choice is mostly infeasible because it is of utter importance to know if an adversary could extract information using a high number of power traces. Works based on highly accurate analog simulations using SPICE simulators [1], [7], [8] and using register transfer level (RTL) simulations [9] for improving simulation speeds have been presented. However, the timely effort of offline power simulations remains a limiting factor for the trace count. The authors of [9] also note that RTL simulations may hide leakage contributors only visible after further design flow steps. To the best of our knowledge no other DPA evaluation approaches exist in literature that are employing hardware emulation techniques.

III. Power Analysis (PA)

Power analysis takes advantage of the fact that the power consumption and the emitted radiation of a given system depends on the workload it executes. This allows for extracting timing and statistical information about the executed algorithm. Therefore it has become a vital source of information for cryptologists and adversaries.

A. Simple Power Analysis (SPA)

In case of SPA the power consumption or radiation profile of a cryptographic system is analyzed in a direct manner, i.e., a low number of traces is searched for specific variations caused by weaknesses of the implementation-under-test. Therefore it is necessary to know at least some details about the used software or hardware implementation.

For example, the usage of branches could cause visible variations depending on the used en-/decryption key. If the different execution paths differ significantly this also can be easily exploited through timing analysis, as the execution time of each path could differ. SPA attacks can be hindered by randomization and general avoidance of data-dependent runtime behavior.

B. Differential Power Analysis (DPA)

DPA is an advanced form of power analysis using statistical methods. It has been published by Kocher et. al. in 1999 [10] and has been proven to be a powerful tool to extract secret key information from cryptographic systems. This technique exploits the fact that the power profile or the radiated electro magnetic field depends on how many logic gates are switching at the same time. Therefore this leaked execution information can be dependent on the used key bits. The techniques originally presented in [10] still required a high amount of power traces to successfully attack a system if basic countermeasures are available. Therefore several advanced approaches have been developed to improve analysis performance such as correlation DPA [11], [12].

IV. Power Emulation (PE)

The power emulation methodology, i.e., the hardware-accelerated power estimation of a given system-under-test, is a promising alternative to simulation-based power profiling approaches. The speed-up of the emulation-based approach in comparison to software simulators is especially useful regarding DPA analysis, allowing for the fast generation of a large number of profile traces.

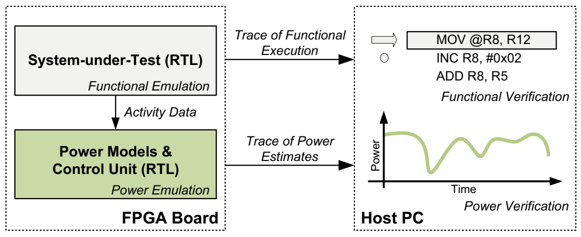


Fig. 1. Power emulation principle: Concurrent functional and power emulation (adapted from [13])

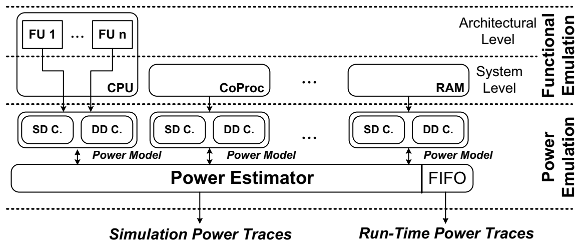


Fig. 2. Extended power emulation unit for DPA evaluations: The unit is monitoring various system components and therefrom derives state- and data-dependent power estimates (adapted from [13])

A. Principle of Power Emulation

The principle of *power emulation* (PE), as initially introduced in [14], is based on state-of-the-art functional emulation that is being augmented with hardware-implemented power models. Thereby, the power estimation process and the functional emulation are executed concurrently during the run-time of the system. The functional and *power emulation* traces are recorded and can be analyzed on a host computer as depicted in Figure 1.

B. Extended Power Emulation Unit

The high-level *power emulation unit* as depicted in Figure 2 and initially introduced in [13] serves as the foundation of this work. The unit utilizes additive linear power macro models as expressed in Equation 1 to generate static and dynamic power consumption estimates for different system components by monitoring power-relevant signals x_i.

$$\hat{P}[t] = \hat{P}_{sta} + \hat{P}_{dyn}[t] = c_0 + \sum_{i=1}^{N} c_i x_i[t] \qquad (1)$$

For the purpose of enabling DPA evaluations, the PE unit has been further extended to monitor power-relevant state as well as data signals and to store a set of state-dependent (SD) as well as data-dependent (DD) power model coefficients for each system component. Originally the PE unit has been analyzing the state of various system components by evaluating selected power-relevant control signals. While this approach is sufficient to determine the power consumption within reasonable accuracy, it is typically not usable for security evaluations because there is no direct relation between power profile and current workset data. Therefore this approach has to be extended with signal switching information of data-dependent architecture elements. To achieve the inclusion of this information an additional pre-processing stage has been added computing an XOR-operation of a data-signal with its state one clock-cycle ago. For our used system-under-test we have chosen to extract this information from both ALU operands and the corresponding result signals. This approach allows to introduce data-dependency into the power consumption model without worsening the *power emulation* result. Compared to the gate-level power simulations the result represents a similar level of accuracy as the power model only employing control signals.

Furthermore, the PE unit has been extended with an internal, configurable FIFO buffer to allow for easy run-time power tracing using our FPGA based evaluation platform. The small memory allows to store the sub-trace of a single encryption run without interference of test control instructions with the cryptographic algorithm. After the code-under-inspection has been finished this FIFO can be safely read and its contents stored to an external memory.

C. Extended Power Characterization for DPA Evaluations

For deriving power models that consist of both state-dependent and data-dependent model coefficients we have further extended the automated power characterization methodology presented in [15]. The power characterization flow as depicted in Figure 3 consists of multiple stages:

1) Generation of training-set data: A set of microbenchmarking applications, covering all components of the system-under-test, is simulated and power-profiled on the given system-under-test using state-of-the-art gate-level power estimation tools. The resulting training-set contains both signal activity as well as power estimation data.
2) Parameter selection: The training-set data is then used to determine power-relevant signals, i.e., power model parameters x_i. In contrast to the completely automated approach considering mainly control-related signals [15] for power modeling, we now also add data signals based on a user-defined configuration. Data signals are preprocessed using a VCD-file analyzer tool to extract only the signal's switching activity.
3) Coefficient fitting: For the parameters selected before, coefficients c_i are determined by means of a model coefficients fitting process using a non-negative linear regression technique. The resulting power model contains both state- and data-dependent coefficients.

V. DPA Evaluation Flow

The proposed DPA evaluation flow is based on the *power emulation* methodology [13], [16] and a power analysis methodology built upon the openly available Open Side Channel Analysis (OpenSCA) kit for Matlab [17]. The hardware-accelerated DPA analysis flow is depicted in Figure 4 and can

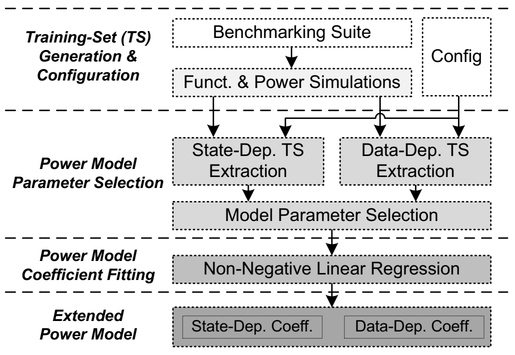

Training-Set (TS) Generation & Configuration
Benchmarking Suite
Funct. & Power Simulations
Config
Power Model Parameter Selection
State-Dep. TS Extraction
Data-Dep. TS Extraction
Model Parameter Selection
Power Model Coefficient Fitting
Non-Negative Linear Regression
Extended Power Model
State-Dep. Coeff.
Data-Dep. Coeff.

Fig. 3. Automated power characterization methodology for DPA evaluations, considering both state- as well as data-dependent power consumption (adapted from [15])

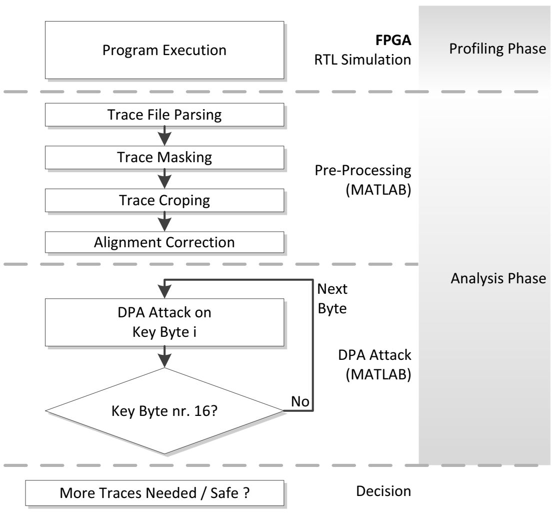


Fig. 4. Hardware accelerated DPA analysis flow

be divided into a *profiling phase* V-A and an *analysis phase* V-B.

A. Profiling Phase

Power profile tracing can be executed in two ways:

- Run-time profile tracing using the extended PE unit
- Power profile tracing using RTL simulation

RTL simulation can be performed very early in the design process. The simulation model generates simple text-style power trace files that can be easily imported into Matlab. In this case minor adjustments are made to the source code to ease sub-profile extraction later on. In our experimental tests we used hard-coded NOP (no operation) instructions to level the power profile for a short period of time.

To generate the same text-style trace file during run-time operation the PE unit has to be augmented with internal tracing memory as described in Section IV-B. After each evaluation run the data saved inside the FIFO has to be read and stored into the test-system's memory.

TABLE I
LEON3 PROCESSOR CONFIGURATION

Operating Frequency [MHz]	40
Instruction Cache Sets	2
Instruction Cache Set Size [kB]	4
Data Cache Sets	1
Data Cache Set Size [kB]	4
MMU TLB Entries	8
MMU Page Size [kB]	4

B. Analysis Phase

After the power profile trace of the executed evaluation program has been recorded, it is parsed for processing in Matlab. This trace covers the whole program execution and therefore has to be divided into its sub-traces containing only one encryption run each. This is done using a two pass approach that is, first, marking the trace separation sections and, second, extracting the sub-traces between these markers. Because of small computational differences between these sub-traces they are cropped to guarantee traces of equal length. Slight delays between these sub-traces can be corrected using cross-correlation.

Finally the stored traces are analyzed using scripts of the OpenSCA toolkit. These have been slightly modified to allow the automatized calculation of all 16 key bytes without further need of user interaction.

VI. Experimental Results

To prove the applicability of our approach we chose a widely used hardware platform and software implementation of the AES cryptography algorithm. For our hardware platform we selected an open source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler [18]. This processor has been synthesized using Xilinx ISE software and tested on the ML507 evaluation board also from Xilinx. The characterization process has been done using gate-level and power simulations using Synopsys PrimeTime provided by our industrial partner. Our operating system tests have been executed using the SnapGear Linux distribution provided by Aeroflex Gaisler. The processor's configuration is summarized in Table I. Our analysis platform consists of MathWorks Matlab 2009b on an six-core 3.2 GHz AMD Phenom-II machine using eight giga-bytes of RAM.

The completed PE characterization process resulted in less than seven percent average and 20 percent root mean square error (RMSE) compared to gate level simulations for all micro-benchmark programs. These results are similar to those of previous versions of the PE unit, using only control signals. The resulting power model consists of 141 coefficients and therefore it was necessary to equip the PE unit with a seven stage pipelined adder tree to reach clock frequency targets. Emulation accuracy could be improved by including more coefficients but at the cost of a more complex hardware design and increased adaption effort for new target designs.

As an exemplary implementation of the AES algorithm we chose a widely known C implementation [19]. This source code provides the user with several customization options to choose between high computation performance and low memory footprint. The slowest possible settings do not use any table functionality and are executed using loops and therefore a high amount of branches. Such an implementation would be easy to attack, as the adversary has a clear view on the current status of the calculation. Therefore we decided to apply high performance settings using predefined SBOX tables. The following options have been activated for this particular implementation: FIXED_TABLES, FF_TABLES, ARRAYS, FAST_VARIABLE.

The target of our experimental evaluations is the AES encryption of randomly chosen 16-byte plaintexts using a 128 bit key. The point of attack is the SubBytes operation for which the hypothetical power consumption will be calculated using the Hamming Weight model.

The whole processor system including the *power emulation unit* has been synthesized using Xilinx ISE 12.3 and tested on the ML507 evaluation board mentioned earlier. For run-time power tracing a 32kSample FIFO has been included using Xilinx Virtex BRAM blocks. For our run-time evaluation system the FPGA board has been equipped with the Linux distribution and the results are saved to the host computer using a network drive (NFS). Performance evaluations have been done using a 152MB memory buffer (4-bytes data width, 2000 traces, 20000 data-points per trace). These 2000 traces have been recorded in 594 seconds resulting in a tracing speed of 3.37 traces per second.

TABLE II
ANALYSIS PERFORMANCE USING 1000 POWER TRACES

Operation	Time [sec]
Tracefile Parsing	766,9
Mask Generation	37,3
Power Trace Masking	152,5
Trace Cropping	17,2
Alignment Correction	4,4
Saving Traces to Container	5,9
Saving Plaintexts to Container	0,1

TABLE III
DPA RESULTS ON THE UNPROTECTED SYSTEM (UNALIGNED TRACES)

Used traces	Corr. Bytes	Corr. Bytes [%]	Time [sec]
200	1	6,25	7,3
300	7	43,75	9,6
400	9	56,25	12,0
500	10	62,5	15,7
600	14	87,5	18,9
700	14	87,5	19,5
800	14	87,5	21,3
850	16	100	22,4

TABLE IV
DPA RESULTS ON THE UNPROTECTED SYSTEM (ALIGNED TRACES)

Used traces	Corr. Bytes	Corr. Bytes [%]	Time [sec]
200	14	87,5	7,6
300	16	100	10,2

TABLE V
TRACING PERFORMANCE: RTL SIMULATION VS. FPGA EMULATION

	Traces	Time [sec]	Speedup
Direct RTL Simulation	100	3850.05	-
FPGA Emulation (w/o OS-overhead)	100	0.0258	149226
FPGA Emulation (w/ OS-overhead)	100	43.13	89.27

A. Unprotected System

First measurement runs are undertaken using an unmodified version of the software AES implementation and LEON3 processor. The performance of the analysis phase is summarized in Table II. Our experimental results show that without any precautions the used AES implementation will leak too much information through its power consumption profile (Table III). Especially if the trace alignment is corrected through cross-correlation the amount of necessary power traces to successfully guess the used key is reduced significantly (Table IV). Therefore the complete 16-byte long key could be extracted using less than 300 traces. It has to be noted that no sophisticated techniques like higher-order or template power analysis techniques have been used to achieve this result. As shown in Table V the FPGA emulation technique provides a significant speedup compared to the direct RTL simulation without any further power estimation steps.

A typical power trace for one encryption run and the result for the key guess of the first key byte is shown in Figures 5.

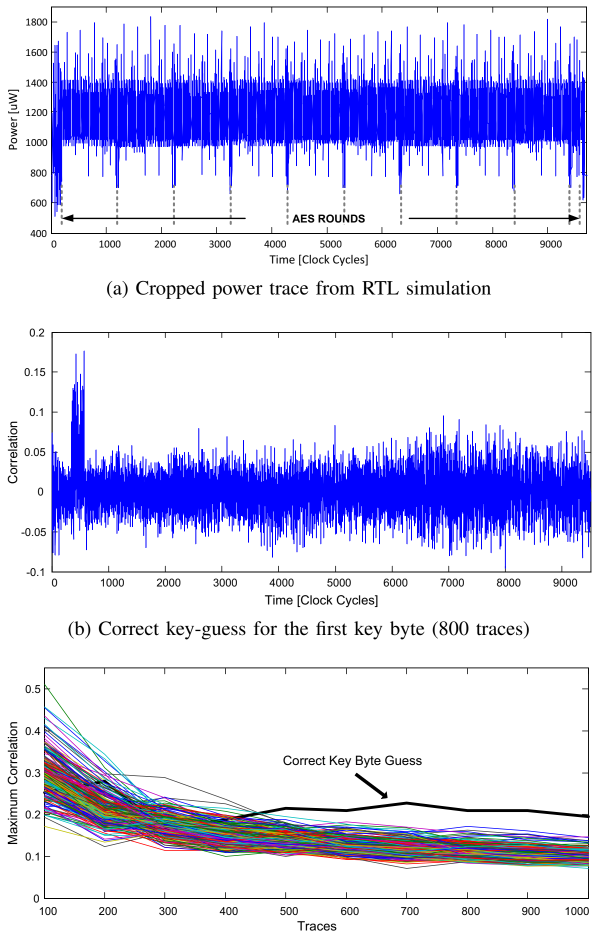
B. Software Countermeasures

A simple and easy implementable countermeasure against DPA attacks would be to introduce randomized elements into the execution. This can be achieved through the random injection of NOP instructions for example. Through the introduction of algorithm-independent instructions the time-relation between algorithm-dependent power contributers and recorded traces is distorted at a small cost of performance degradation. For this proof-of-concept example we implemented a pure software solution in which random values are selected before the evaluation for each test-run. NOPs are inserted after each encryption run to reduce trace correlation.

The maximum number of evaluated traces was 3000 without resulting in exposed key bytes. Therefore the effort to extract secret information has been increased sufficiently to protect against our DPA attack setup.

C. Hardware Countermeasures

Similar to the software approach shown in Subsection VI-B the DPA result is influenced by reduction of inter-trace correlation. This is achieved by run-time augmentation of



(a) Cropped power trace from RTL simulation

(b) Correct key-guess for the first key byte (800 traces)

(c) Correlation for all key possibilities depending on trace amount

Fig. 5. Tracing and DPA analysis results for an unprotected system

the processor's clock signal. Depending on the steepness of existing power peaks in the power profile the clock signal is masked.

We recorded up to 3000 traces without being able to extract the used key. The clock signal manipulation while being easy to implement comes with a high increase in execution time (9600 vs. 17400 clock cycles). The reduction of the strength of this clock manipulation approach would lead to higher correlation and therefore a good balance between performance and efficiency has to be found.

VII. Conclusion

In this paper we have shown that run-time *power emulation* hardware accelerates the analysis of software implementations of the AES cryptographic algorithm for their sensitivity to power analysis attacks. This opens several new possibilities to designers of embedded systems for cryptographic purposes and provides them with a high performant analysis tool.

We showed that the usage of *power emulation* techniques not only allows to analyze a high amount of power traces during the design phase, but also how this information can be instantly used to improve system software and hardware. Most of the used and proposed analysis flows are highly automatized and therefore do not imply additional effort for the system designers.

The proposed hardware could also be used for run-time power analysis in complex system-on-chip designs. Explorations in this direction will be part of our future work.

Acknowledgment

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749. We would also like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

References

[1] F. Regazzoni, S. Badel, T. Eisenbarth, J. Grossschaedl, A. Poschmann, Z. Toprak, M. Macchetti, L. Pozzi, C. Paar, Y. Leblebici, and P. Ienne, "A Simulation-Based Methodology for Evaluating the DPA-Resistance of Cryptographic Functional Units with Application to CMOS and MCML Technologies," in *IC-SAMOS 2007*, 2007, pp. 209 –214.

[2] V. Tiwari, S. Malik, A. Wolfe, and M. Tien-Chien Lee, "Instruction level power analysis and optimization of software," *The Journal of VLSI Signal Processing*, vol. 13, no. 2, pp. 223–238, 1996.

[3] K. Tiri and I. Verbauwhede, "Simulation models for side-channel information leaks," in *DAC 2005*. ACM, 2005, pp. 228–233.

[4] D. Shumov and P. L. Montgomery, "Side Channel Leakage Profiling in Software," in *COSADE 2010*, February 2010.

[5] J. den Hartog and E. de Vink, "Virtual Analysis and Reduction of Side-Channel Vulnerabilities of Smartcards," in *Formal Aspects in Security and Trust*. Springer, 2005, pp. 85–98.

[6] C. Thuillet, P. Andouard, and O. Ly, "A Smart Card Power Analysis Simulator," *CSE 2009*, vol. 2, pp. 847–852, 2009.

[7] H. Li, A. Markettos, and S. Moore, "Security evaluation against electromagnetic analysis at design time," in *HLDVT 2005*, 2005, pp. 211 – 218.

[8] F. Regazzoni, A. Cevrero, F.-X. Standaert, S. Badel, T. Kluter, P. Brisk, Y. Leblebici, and P. Ienne, "A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions," in *CHES 2009*. Springer, 2009, pp. 205–219.

[9] M. Bucci, R. Luzzi, F. Menichelli, R. Menicocci, M. Olivieri, and A. Trifiletti, "Testing power-analysis attack susceptibility in register-transfer level designs," *IET 2007*, vol. 1, no. 3, pp. 128–133, 2007.

[10] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO 99*. Springer, 1999.

[11] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *CHES 2004*, pp. 135–152, 2004.

[12] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Verlag, 2007.

[13] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss, "An emulation-based real-time power profiling unit for embedded software," in *SAMOS 2009*, 2009, pp. 67–73.

[14] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *DAC 2005*, 2005, pp. 700–705.

[15] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid, "Automated Power Characterization for Run-Time Power Emulation of SoC Designs," in *DSD 2010*, 2010, pp. 587–594.

[16] ——, "Accelerating Embedded Software Power Profiling Using Run-Time Power Emulation," in *PATMOS 2010*. Springer, 2010, pp. 186–195.

[17] E. Oswald, "OpenSCA, An open source toolbox for Matlab," December 2010. [Online]. Available: http://www.cs.bris.ac.uk/home/eoswald/opensca.html

[18] Aeroflex Gaisler, "LEON3 Processor," December 2010. [Online]. Available: http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53

[19] B. Gladman, "Implementations of AES (Rjindael) in C/C++ and assembler," December 2010. [Online]. Available: http://gladman.plushost.co.uk/oldsite/cryptography_technology/index.php

ARTICLE IN PRESS

Microprocessors and Microsystems xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Microprocessors and Microsystems

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

Case study on multiple fault dependability and security evaluations

Johannes Grinschgl [a,*], Armin Krieg [a], Christian Steger [a], Reinhold Weiss [a], Holger Bock [b], Josef Haid [b], Thomas Aichinger [c], Christiane Ulbricht [c]

[a] Institute for Technical Informatics, Graz University of Technology, Inffeldgasse 16/I, Graz, Austria
[b] Infineon Technologies Austria AG, Design Center Graz, Babenbergerstrasse 10, Graz, Austria
[c] Austria Card Plastikkarten und Ausweissysteme GmbH, Lamezanstrasse 4-8, Wien, Austria

A R T I C L E I N F O

Article history:
Available online xxxx

Keywords:
Fault injection
Fault emulation
Multi-bit faults
Saboteurs

A B S T R A C T

The increasing level of integration and decreasing size of circuit elements leads to higher probabilities of operational faults. More vulnerable electronic devices are also more prone to external influence from energizing radiation. Additionally, the concerns of chip designers include not only the natural causes of faults but also the misbehavior of chips due to "planned" attacks, as, for example, in critical security applications. In particular, smart cards are exposed to complex attacks through which an adversary attempts to extract knowledge from secured systems by provoking undefined states. These problems increase the need to test new designs for their fault robustness.

This paper presents a case study on fault injection strategies. An in-system fault injection strategy for automatic test pattern injection by enabling the emulation of fault effects on the circuit level is introduced. Second, an approach is presented that provides an abstraction of the internal fault injection structures to a more generic high-level view. Through this abstraction, it is possible to help the operating system designer test a product against different fault effects without knowing how to produce this effect by a fault attack. Therefore, we implemented a modular fault injection controller that is located along with the system under test on the emulator platform.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The continuing success of the semiconductor industry with regard to downscaling structures has led to highly integrated but also highly sensitive devices. External radiation effects and thermal and electric degradation have become common problems for the dependability of a system [1].

Through these effects, transient or even permanent faults are introduced, which leads to a change in the system behavior. These faults occur randomly, unlike faults resulting from so-called fault attack scenarios. In this case, an attacker deliberately injects faults into a system to change the system behavior. Without dedicated precautions, such attacks are easy to implement [2].

In recent years, intensive research has introduced several different tools to simulate or emulate possible fault scenarios during the design phase. In particular, fault emulation proved to be a very effective way of testing systems under the influence of fault sources. The platform usually used to emulate such a faulty system is a field programmable gate array FPGA because of its flexibility. An example of such an FPGA fault injection platform is shown in Fig. 1. There are different ways to inject faults into circuits. One is the use of partial reconfiguration features of the FPGA [3], but this design approach heavily limits the platform choice because there are only a few candidates, such as the Virtex families from Xilinx [4]. Another way is to instrument the given circuit either with manipulated logic elements or with integrated controllable fault elements. The latter case can distinguish between saboteurs and mutants [5]. Saboteurs are small circuit elements that do not affect the system behavior under normal conditions. If activated, they directly inject faults into the targeted submodule by disturbing the internal signals. To disturb signals, saboteurs must be placed between their source and their sink. Mutants are modified submodules that also do not affect system behavior under normal conditions but, if activated, behave like a faulty version of the original. To simulate or emulate fault attacks with mutants, the submodule must be replaced by a mutated submodule.

To accomplish such a test setup, it is necessary to have access to the hardware description or a standardized test interface. Such a test interface could consist of test chains [6]. Another important step in creating an effective fault injection platform is the selection of a proper fault model. For dependability evaluations, a single event upset (SEU) fault model is often sufficient. If the faults are caused by radiation or degradation, it can be safely assumed that only a single random fault will occur at a time [7]. In contrast, security evaluations consider intentional faults. Therefore, it is possible

* Corresponding author.
E-mail address: Johannes.Grinschgl@tugraz.at (J. Grinschgl).

0141-9331/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2012.05.016

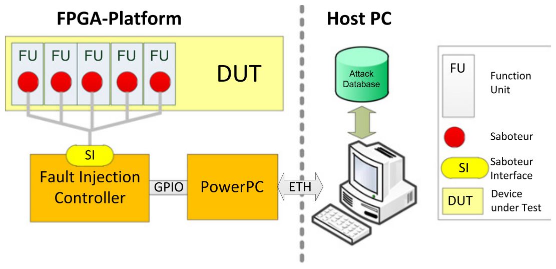


Fig. 1. Schematic view of the proposed fault injection system (obtained from [27]) .

that an attacker introduces several faults at once to achieve the desired effect of secret exposure.

This type of multi-bit fault model results in much more complex fault relationships than SEU models because the time and space of such faults cannot be assumed to be random. This model must reflect the interdependency of several SEUs. Thus, in dependability analyses, similar types of sensitive sub-circuits will fail at the same time. In security evaluations, this dependency is defined by the attack scheme of the adversary.

Finally, the controlling element itself must be designed very carefully. The scope ranges from a very simple implementation completely controlled by an external source such a personal computer to very sophisticated integrated designs. Simple implementations have the advantage of being very adaptable to system changes, but they are also highly dependent on the communication interface used. Therefore, they can be considered slow. An implementation based on a PCI interface is shown in [6]. Sophisticated solutions allow for very high fault injection rates but are difficult to adapt to system changes [8]. In recent years, implementations of fault injection emulation controllers have been either simple and limited by slow interface performance or highly complex and therefore not portable from one design to another.

In this paper, we propose a modular fault injector (MFI) that combines the advantages of a simple portable design and fast, highly complex implementations. It also helps to provide a common platform for fault injection campaigns on different target architectures. Another point that has often been neglected in previous work in this field is the consideration of fault attacks.

Finally, the consideration of multi-bit fault patterns instead of simple SEUs can be seen as a main contribution of this work. This consideration is especially important for security evaluations. This fully synthesizable controller can also be used as an online-testing implementation if desired. The main goals of this work can be summarized as follows:

- Fault effect modeling for software development.
- Fully modular fault injector design.
- Multi-bit fault injection to support fault attack emulation.

This paper is structured as follows. Section 2 briefly describes the state of the art of work related to emulated fault injection using integrated controllers and different reconfiguration techniques. In Section 3, the design of the fault injection controller is described. Section 4 presents some of the experimental results of this MFI by means of the LEON3 processor [9]. Finally, conclusions are drawn in Section 5.

2. Related work

The introduction of fault emulation on FPGA platforms has led to several publications concerned particularly with the emulation of SEUs in space applications. Approaches using direct circuit manipulation such as the solution proposed in this paper must be distinguished from techniques using the reconfiguration features of certain FPGAs.

The former possibility can again be divided into two subcategories. One uses specialized hardware units to instrument saboteurs or mutants as shown in [8]. The second variant uses available processor cores for fault injection automation similar to the solution provided by [10]. This solution promises a high emulation performance because of fast on-chip communication channels, but it is more difficult to port to other platforms.

In recent years, the partial and complete runtime reconfiguration of FPGAs became a common technique to implement fault injection in a flexible way. Reconfiguration is possible on special FPGA series using an external communication interface to directly feed different fault configurations into the FPGA, as shown in [11–13]. If this external interface is too slow, it is also possible to use existing processor resources for the reconfiguration task, as presented in [14,15]. While the reconfiguration approach is tempting (and has been used by several research groups), it also limits the maximum reachable fault injection performance and the designer's platform selection. An approach similar to the reconfiguration approaches is presented in [16], in which the synthesized netlist is augmented to preserve the original structure of the system.

In the field of security evaluations, the authors of [17] used a proven fault injection platform presented in [18]. This investigation confirms the strong need for multiple fault injection campaigns in the design process. However, in their experiments, only a small fault multiplicity (six) was used to prove this point. In [19], a system demonstrating the use of multiple emulation platforms to run simultaneous fault injection campaigns is described. This parallelism increases the emulation speed.

3. Modular fault injector

For this case study, a fault injection method is required. Therefore, a modular fault injection controller concept and its implementation are presented in this section. The following requirements summarize the main drivers behind the development of this modular fault injector (MFI):

- Effect modeling for software development.
- Support for fault patterns to support multi-bit fault injection.
- Multi-mode saboteur support.
- Scalable interfaces to support automatic saboteur placement.
- Standardized communication interface.
- Internal memory for automatic fault injection.

To support the easy application of the fault injection system to a new design under test, a standardized communication interface is needed. The General Purpose Input/Output (GPIO) communication interface enables the developer to use the proposed controller in a wide selection of different architectures. The GPIO interface consists of pins that can be configured via software commands. These pins allow for controlling the MFI without disturbing the device-under-test. Extensive fault injection campaigns are only possible using a large number of active saboteurs. It is not possible to route such a large number manually; therefore, internal interfaces must be scalable to enable automated injection processes. An effective separation of the design and evaluator tasks can only be guaranteed through a generalized view of the fault injection system, which is accomplished using so-called fault patterns, high level representations of the underlying saboteur distribution. These

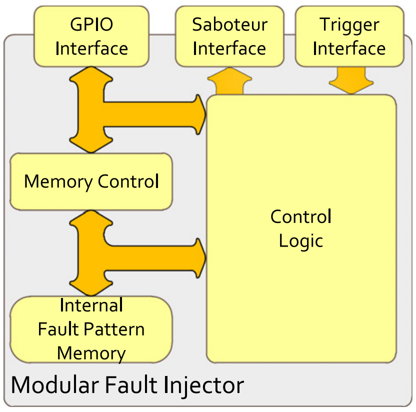


Fig. 2. Schematic view of the modular fault injector (adapted from [27]).

patterns are also necessary for efficient physical fault attack emulation.

High-speed automated operation is enabled through a flexible internal memory system. In the design phase of the proposed fault injector, future advances of the evaluation platform must be considered, which include support of large fault patterns and burst loading from the bus-system. Different fault and attack scenarios call for different saboteur types. To simplify the reconfiguration process, different saboteur modes are provided by a single flexible saboteur design.

A basic block diagram of the proposed fault injection system is shown in Fig. 1. The flow consists of the fault injection controller, which specifies the attack scenario; the saboteurs, which disturb signals; and a PowerPC, which is used as the interface to the PC. The PowerPC is an optional component because the interface of the MFI can be easily configured. As described in the following sections, the MFI is controlled over a GPIO interface. This interface can also be controlled from different devices, e.g., for Smart card fault emulation, and during the development process of the operation system, the MFI can be controlled with APDU commands.

3.1. Fault injection controller

The fault injection controller is the main part of the MFI. It controls the mode and the activation time of the saboteurs. As shown in Fig. 2, the fault injection controller consists of two interfaces. The first is the saboteur interface. This interface is a bus where, for each saboteur, an activating signal is included. It also contains mode signals to control the mode of the saboteurs. The second port is the General Purpose Input/Output (GPIO) port. This interface is used to control the fault injection controller. The internal fault pattern memory is filled via the GPIO port. This memory is used to specify when each attack pattern is activated.

Due to the GPIO interface of the MFI, it is possible to automate fault injection campaigns. The easiest way, when using a Virtex5 FXT FPGA, is to control the MFI via GPIO commands through the FPGA-internal PowerPC. Using this solution, a simple, generic software interface to the generic hardware interface is provided, which has the advantage that the test-engineer does not need specific knowledge about the fault injection system itself. Especially in the case of security evaluations, the evaluation engineer will be mostly concerned with the proper mapping of a real attack scenario to the saboteur matrix inside the system of interest. The GPIO interface of the MFI can be easily changed to every other interface, which allows the writing of the internal memory of the MFI. This flexibility makes the whole flow independent of the test environment.

3.1.1. Implementation

The fault injection controller is split into five main parts.

- **Saboteur interface:** The saboteur interface was implemented to allow for easy expandability to support several hundreds of saboteurs. The size of the saboteur interface bus is equal to the number of saboteurs plus the mode signals to control the mode of the saboteurs. Another method to control the saboteurs is the scan-chain approach. The disadvantage of the scan-chain approach is that multiple clock cycles are required for the configuration of the saboteurs.
- **GPIO interface:** The GPIO interface is the interface of the MFI to the controlled processor. All attack scenarios can be loaded into the MFI via this port.
- **Memory control:** The memory control is used to place the fault patterns into the internal fault pattern memory. It also generates signals for the control logic if a special command is sent via the GPIO port.
- **Internal fault pattern memory:** This memory stores the fault patterns, which define the attack scenario.
- **Control logic:** The control logic activates the saboteurs depending on the information stored in the internal fault pattern memory.
- **Trigger:** The MFI also has a trigger source, which can be used to synchronize the fault injection with the current state of the executed software.

3.2. Saboteurs

To model a wide selection of possible faults, we introduce a configurable saboteur. According to [20,21], saboteurs can be categorized into several different types. Depending on their location, they can be differentiated between serial and parallel saboteurs. Depending on the directionality, a division can be made between uni- and bidirectional ones. Finally, depending on their complexity, simple and complex saboteurs can be distinguished. The saboteur used in this work can be classified as a unidirectional, serial, simple saboteur. Thus, it only works in one direction, it is located directly in the connection signal, and it affects only one port at the input and output side. The supported fault models of such an injection element are shown in Table 1.

The proposed saboteur can be used to inject faults into single bit lines or even complete buses, which allows the emulation of bus attacks with, e.g., unfocused lasers or influence by strong external energy sources. The first four saboteur configuration modes are equivalent to direct circuit modifications. Bit-flips could also be forced by short, intensive, external pulses. Delay faults emulate circuit behavior in the case of operating voltage changes. A schematic block diagram and fault effect visualization of the proposed single-bit saboteur is shown in Fig. 3.

Table 1
Saboteur operation types and their description.

Saboteur mode	Fault type	Description
Stuck-at-zero	Permanent	Signal value of '0' until reload
Stuck-at-one	Permanent	Signal value of '1' until reload
Indetermination	Permanent	Undefined signal state until reload
Bridging fault	Permanent	No output propagation until reload
Negation of input	Permanent	Undefined signal state until reload
Bit-flip	Transient	Output inverts input for one cycle
Artificial delay	Transient	Input to output propagation delay

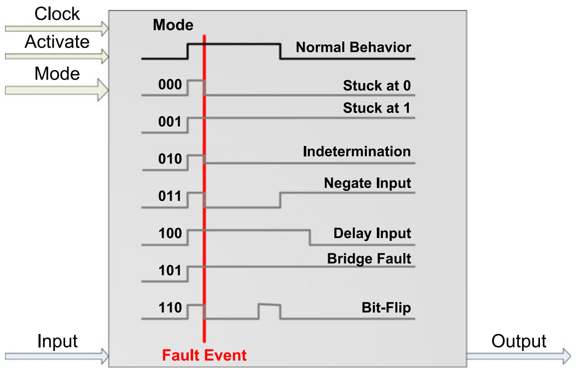


Fig. 3. Block diagram of the proposed saboteur element (obtained from [27]).

3.2.1. Implementation

The complexity of the implementation of the saboteurs depends on the features a saboteur supports. Therefore, the required number of slices for one saboteur mainly depends on their complexity. If the interface of the saboteurs is not changed, only the architecture of the saboteur must be modified to support more features. Thus, once the saboteurs are placed, the evaluation engineer can adapt the saboteurs by changing only one file, which makes the saboteur placement concept very flexible.

3.3. Fault pattern support

To evaluate a device-under-test for its fault attack sensitivity, it is important to know the exact location of security relevant silicon regions. Because the system is designed for fault effect modeling to test security features during software development, the required locations to place saboteurs are limited. Important locations can be memory interfaces, inputs and outputs of the system or inputs and outputs of calculation modules. At these locations, saboteurs must be placed by an evaluation engineer. A fault pattern approach is used for an effective mapping of saboteurs to their corresponding fault locations. Of course, not every possible pattern combination must be transmitted to the MFI. Every pattern used represents a very likely fault effect. For example, memory data at a specific address have been manipulated. Where exactly the attack was injected does not matter; only the effect that can be observed by the software is important for the operating system designer. For hardware evaluation, other methods must be used. This approach allows the user to check that the software security features developed work as intended, independent of how the fault effect is produced. The basic concept of mapping the fault pattern with the physical implementation is shown in Fig. 4.

The encoding of the fault pattern is performed by an array with *x* columns and *y* rows. Each element represents one saboteur or an area where no saboteur is placed. To define a special attack pattern, only an array where the active saboteurs are marked must be transmitted to the MFI.

3.3.1. Implementation

In this example, two methods of pattern generation are used to show the function of the fault MFI. The first one is a random number generator to show the functionally of the MFI. The second method is a pattern defined by a test engineer, which can be used for fault effect production. The complexity of such a pattern generation does not increase the required slices of the MFI because the patterns are not generated directly in the MFI.

3.4. Automatic placement of saboteurs

For a large number of saboteurs, an automatic saboteur placement approach is required because manual saboteur placement is a very time consuming and error prone process. In [20], a theoretical method is discussed for placing saboteurs automatically into a VHDL design. Our approach works very similar to that approach. In Fig. 5, a schematic example of automatic saboteur placement is shown. The saboteur placement tool is based on the vMAGIC VHDL parser library [23]. VMagic is a Java API that can parse and adapt VHDL code automatically. The flow can automatically add hundreds of saboteurs [22]. Then, the VHDL code is automatically adapted with saboteurs and the fault injection controller. A method to modify the VHDL code automatically is shown in [24].

3.4.1. Implementation

For this paper, the automatic saboteur placement tool was used to place the saboteurs into a critical signal in the design, which improves the very error prone and time consuming process of manual placement of the saboteurs.

3.5. Attack scenario

As shown in Fig. 6, the attack flow is based on a golden model run. The golden model runs from one reset to the next reset of the DUT. All of the golden model information is stored (e.g., the output, none volatile memory (NVM), and timing).

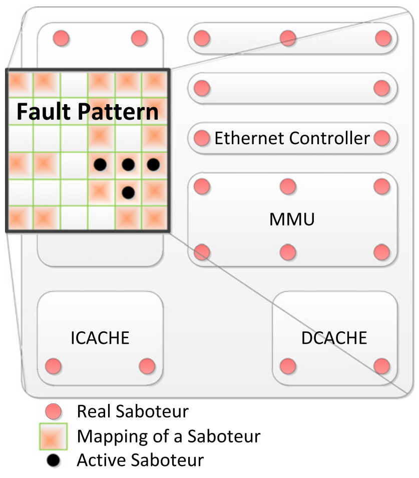


Fig. 4. Schematic view of an extracted fault pattern (obtained from [27]).

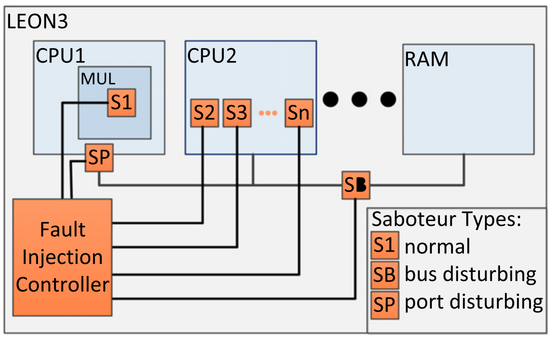


Fig. 5. Automatic routed saboteurs (adapted from [22]).

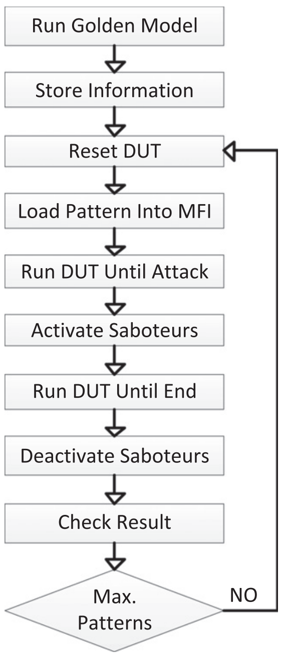


Fig. 6. Flow diagram shows a typical fault injection run (adapted from [27]).

The next step is to reset the whole system. Then, all of the attack patterns are stored in the internal fault pattern memory via the GPIO interface. In addition, the attack mode must be transmitted via GPIO. The mode is used to define the attack type of the saboteurs (e.g., bit-flip). After a pre-determined time, the specified fault attack will be injected, and then the saboteurs are activated depending on the fault pattern. Now, the faulty DUT will run until it is reset or a pre-defined timeout is reached. This timeout is required because the device could run into an infinite loop after the fault injection.

After the fault injection run is finished, the saboteurs are deactivated, and the result is compared to the results from the golden model run. If the results are not equal, a fault analysis process must be launched. This procedure is continued until all attack patterns are tested.

The fault pattern generator creates attack patterns for the fault injection. In this example, only a random number generator is used as the pattern generator to show the function of the fault MFI. The complexity of such pattern generation does not increase the required slices of the MFI because the patterns are not generated directly in the MFI.

4. Experimental results

To prove the effectiveness of our approach, we chose to implement the proposed controller on a widely used platform, the LEON3 SPARC V8 conformant processor from Gaisler Research [9].

The test setup is implemented on a Virtex5 FPGA using the Xilinx ML507 evaluation board. Simulation results are obtained using Modelsim software, which is part of the ISE software package provided by Xilinx. The fault injection setup is configured as shown in Table 2.

4.1. LEON3 platform setup

The LEON3 is configured for a single-core configuration using the default platform settings for this particular evaluation board (ML507). The MFI is configured via GPIO and can be accessed using the PowerPC.

4.2. Saboteur configuration

The random approach of fault injection proposed in this publication can find global dependencies between local fault occurrences. Of course, with an increasing number of saboteurs, the number of possible combinations increases until an analysis of the results is not possible in a reasonable time frame. Therefore, a small number of saboteur locations were chosen for these fault injection experiments.

4.3. Simulation results

To ensure the correct behavior of the fault injector and its saboteur, the whole LEON3 system are thoroughly simulated using the Modelsim software package. The simulation includes not only the fault injection process but also start and calculation instructions. The results of these injection campaigns are shown in Table 3.

The pattern injection rate is constant for patterns smaller than 32 because of the working principle of the MFI. For larger patterns, additional GPIO transfers are required. However, with increasing pattern size, the amount of concurrently injected faults increases accordingly.

4.4. Proof of fault models

To show that all possible fault models of the saboteur work as intended, a simple program is attacked. The program attacked is a simple program that sends five bytes via the UART to a host PC. On the UART interface, saboteurs are placed and activated during the send process. Table 4 shows the effects that can be reproduced with the saboteurs. As shown, all of the effects can be reproduced in software as well on the FPGA.

4.5. VHDL synthesis results

To estimate the necessary FPGA area requirements, the synthesis of a typical platform configuration has been performed. The results of several synthesis runs using different saboteur configurations are shown in Table 5.

The MFI and its saboteurs have a negligible effect on the size of the resulting synthesized design. Basing on the routing results, it should be possible to implement a large number of saboteurs on this FPGA platform. These results show that the saboteurs and the MFI produce an overhead to the system. The PowerPC is an op-

Table 2
Fault injection setup.

Saboteur type	Fault mode	Fault target type	Fault injection targets
Single-bit	Bit-flip	Control logic	Integer pipeline
			Cache controller
			Register file
			Multiplier unit
			Divider unit

Table 3
Simulated fault injection performance.

Saboteurs	Inj. patterns	Time (s)	Patterns (s)
8	256	274	0.93
16	256	282	0.91
24	256	286	0.90
32	256	299	0.86

Table 4
Saboteur fault model simulation and emulation results.

Saboteur mode	Simulation	Emulation
Stuck-at-zero	ok	ok
Stuck-at-one	ok	ok
Indetermination	ok	ok
Bridging fault	ok	ok
Negation of input	ok	ok
Bit-flip	ok	ok
Artificial delay	ok	ok

Table 5
VHDL synthesis results.

Saboteurs	Look-up-tables	Overhead (%)	Slices	Overhead (%)
0	14897	–	10423	–
8	14950	0.36	10513	0.86
32	15107	1.41	10642	2.10
96	15194	1.99	10716	2.81
170	15244	2.33	10774	3.37
326	15478	3.9	11088	6.34

Table 6
Emulated fault injection performance.

Saboteurs	Inj. patterns	Time (s)	Patterns (s)	Pattern blocks (s)
8	100 M	46.76	2.17 M	2.17 M
32	100 M	46.76	2.17 M	2.17 M
96	100 M	114.48	873.5 k	2.17 M
170	100 M	156,4	639.4 k	2.17 M
326	100 M	282.16	354.4 k	2.17 M

tional block to control the MFI. This controller is only used to control the GPIO of the MFI. The GPIO can be controlled directly by a host PC or any other tool.

4.6. Fault injection performance

In this subsection, the results of several fault injection campaigns are presented to show that higher structural flexibility does not come with disadvantages concerning emulation speed. It also must be mentioned that such a multi-bit injection campaign leads to complex results in system behavior, and therefore, the analysis step will most likely be the slowest link in the emulation chain. The results are summarized in Table 6. The number of saboteurs has an influence on the number of patterns that can be injected per second. The reason is that the GPIO interface only support the transmission of 32 bits of the pattern simultaneously. If the size of the pattern is increased, the time required for the pattern transmission also increases. The last column shows that the transmission of one pattern is interdependent of the number of saboteurs. It is not necessary to send the whole pattern for each attack. If only one pattern block must be changed, only one GPIO transmission is required. Only the maximum number of faults is important because not all of the saboteurs are activated. The more important numbers are the patterns injected per second. The speed of the pattern injections per second is not independent of the number of saboteurs. However, the time required for the transmission of one fault pattern is constant.

The results presented in Fig. 7 shows the maximum speed that the fault pattern can be changed per second. This number is important if an attack is emulated by a laser moving over the chip. Thus, it is possible to change the fault effect in the range of 300 k–2 M times per second. If one fault per run should be injected, the whole test requires the run time of the program that should be tested in addition to the configuration time of the MFI.

4.7. Case study: Attack on RSA authentication

The MFI is used to show a case study of a fault attack on RSA authentication [25]. The RSA authentication process signs a message with a private key. To generate the signature, the RSA algorithm uses multiplications. In [25], it is concluded that fault attacks on the multiplication unit can be used to attack the RSA algorithm. For this fault attack, the authors assume that only one bit of the multiplier output switches. In this case study, we show that our approach can emulate such fault attacks exactly.

For this test, we chose the LEON3 processor in a single core configuration. For the attack target, the output signals and one of the operand registers of the multiplier have been chosen. Because only the fault effect of this one-bit result attack should be reproduced, no hardware error detections methods are implemented in the LEON3 processor.

An overview of the test environment is given in Fig. 8. Listing 2 shows the pseudo code describing the fault injection flow. The saboteur placement methodology is described in [22]. The FPGA-internal PowerPC runs a Linux kernel to allow programming of the fault injection flow in a comfortable way. The fault injection control software runs on the PowerPC. The host PC evaluates the results of the fault injection. During this attack, the saboteurs are configured in bit-flip mode.

As shown in Listing 1, the code that is executed on the LEON3 is a simple program that only calculates one multiplication and send the result back to the host PC. This program should only be used as a proof of concept.

The results are shown in Table 7 and Fig. 9. The MFI can emulate a fault attack on the multiplier interface. Because the MFI supports a large number of saboteurs, it is possible to place the saboteurs at not only the output: it is also possible to test also the effect of a fault injection to the input of the multiplier. If the operant 1

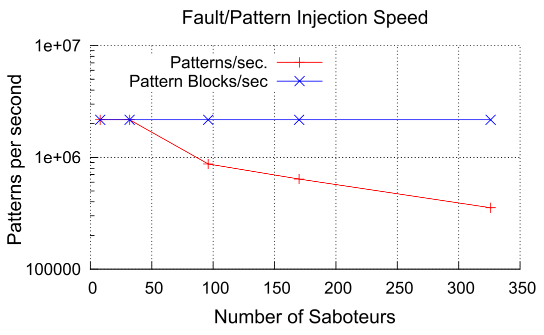


Fig. 7. Speed of the fault injection depending on the number of saboteurs.

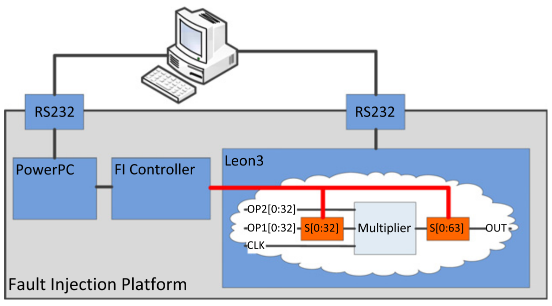


Fig. 8. Overview of the test environment (adapted from [27]).

```
main(){
  init_system();
  output = a*b; //attacked operation
  print(output);
}
```

Listing 1. Pseudo code of the LEON3 software.

```
run_golden_model();
store_all_information();
for(i=0;i<n_patterns;i++){
        reset_dut();
        load_pattern(pattern[i]);
        run_DUT(pattern_time[i]);
        activate_saboteurs();
        run_DUT(max_time);
        deactivate_saboteurs();
        check_result();
}
```

Listing 2. Pseudo code of the fault injection flow.

Table 7
Number of faults detected depending on the attack pattern.

Attack target Number of saboteurs	OP1 1	OP1 2	OP1 3	OP1 4	Out 1
No. of bit-flips					
0	37	37	37	37	0
1	208	0	0	0	5000
2	275	365	129	59	0
3	477	428	532	224	0
4	676	624	478	646	0
5	899	836	862	704	0
6	942	862	762	852	0
7	714	686	787	817	0
8	369	529	580	604	0
9	226	337	432	487	0
10	105	170	221	319	0
11	40	82	99	167	0
12	20	22	56	58	0
13	9	13	13	24	0

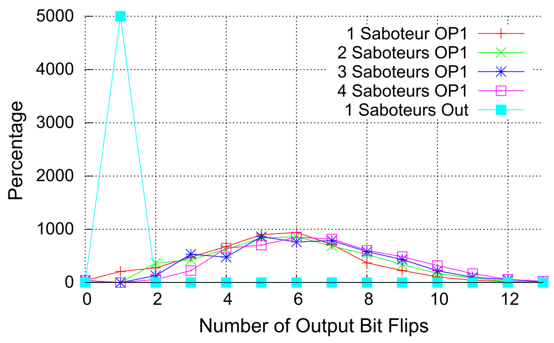


Fig. 9. Influence of saboteurs on the multiplication unit. Bit-flips of the output depending on the number of active saboteurs at the input/output.

(OP1) is attacked, the number of output bits switches depends on which bit of the input is attacked. A similar effect can be reproduced by attacking multiple bits at OP1. If exactly one bit at the output should be disturbed, the simplest method is to attack the output (OUT) of the multiplier with exactly one saboteur.

This emulated fault reproduces a fault attack on RSA authentication where exactly one output bit of the multiplication must be disturbed [25]. To test whether the operating system can detect such an attack, it makes sense to place saboteurs at the output of the multiplier and manipulate exactly one bit of the result.

4.8. Case study: Long-time test for light attack emulation on the memory of a smart card

The MFI can also be used for tests with long durations. This test shows how the MFI can be used in systems development to improve security. As an example, the estimated effect of a light attack on a smart card memory is emulated [26]. In particular, the effect of permanent faults injected into the memory should be evaluated. Therefore, security relevant memory regions must be attacked every clock cycle.

Fig. 10 shows that only some memory regions at special times are security relevant. The relevance of the different memory regions must be defined by the evaluation engineer. Some criteria for the engineer by which code regions should be tested can include regions that.

- Contain final or intermediate results of an security relevant function.
- Contain keys of an encryption process.
- Contain, for example, an AES S-Box.

This reduction is required because attacking every memory cell is too time-consuming. In Eq. (1), the number of tests required is calculated. If the critical memory address section is 100 bytes, the critical regions consist of 1 k clock cycles, and the number of tested patterns per memory address is 10, we obtain an overall number of 1 M tests. Assuming that the time required for one run is 500 ms, then the whole test would require approximately 6 days.

$$N_{tests} = N_{memoryaddresses} * N_{clkcycles} * N_{attackpatterns} \qquad (1)$$

Fig. 11 shows the structure of the fault emulator for this fault injection campaign. For this test, the MFI is placed directly into the Smart Card system. The MFI is controlled via a serial interface, which can also be used to control the smart card and allows us to control the MFI without adding an additional port to the smart card emulation system.

For this test, we used software that requires 150 k clock cycles for a full execution. The security-relevant region of this code is approximately 1000 clock cycles long. During these cycles, an authentication process is performed. To test whether the operating system can cope with light attacks on the memory, we choose 100 attack points within security relevant regions. We also reduced the tested memory section to 1536 bytes because only these bytes are

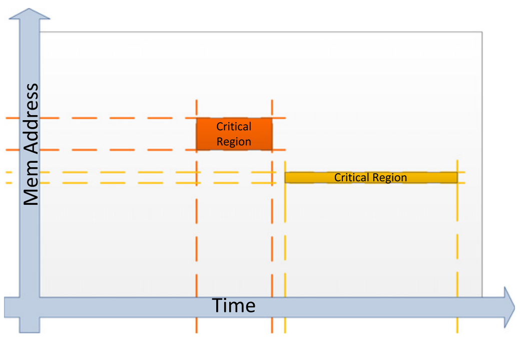


Fig. 10. Critical memory regions of a smart card system.

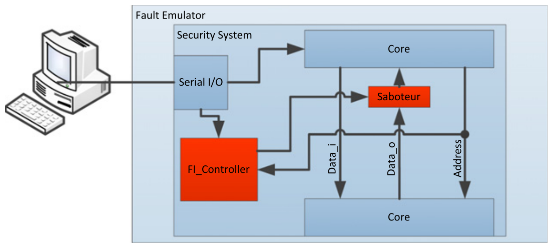


Fig. 11. System modifications for the RAM fault emulations.

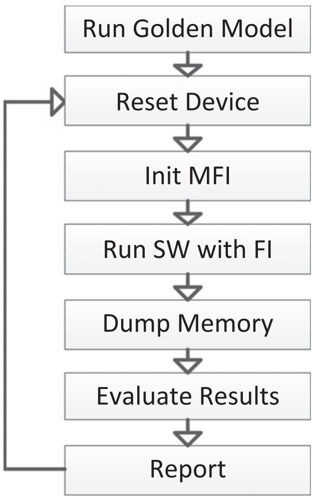


Fig. 12. RAM fault emulation flow.

Table 8
Results of the light attack emulation on the memory.

	N	%
Total number of injections	156672	100
Memory cell never accessed	121570	77.6
Fault injected	35102	22.4
No effect	30373	19.4
Fault detected by smart card	4729	3.0

security-relevant. To emulate a permanent fault, we set the mode of the saboteurs to stuck-at-one. This leads to a total of 157 k tests (see Eq. (2)).

$$N_{tests} = 1536 * 102 * 1 = 156672 \quad (2)$$

The flow of this test is shown in Fig. 12. The test flow can be split into seven main parts:

Table 9
Fault injection performance compared to previous results.

Approach	Clock cycle (ns)	Inj. faults	Time (s)	Inj. speed 1/(s)	No. inj. speed 1/(s)
[6]	50	100 k	83	1205	603
[12] AE	16.8	129.75 M	698	185888	276619
[12] PR	18.97	10 k	1014	9.86	13
[14] DPR	10	–	–	12987	32468
[14] PRR	10	–	–	414.94	1037
This work	25	100 M	46.76	2.17 M	2.17 M

- **Run a Golden Model:** The golden model run is required as reference to evaluate whether an attack was successful or not. During the golden model runs, all output values and the content of the memory are stored.
- **Reset Device:** The reset is required to start the execution at the same point for each run. This reset is the reason that the execution time of a fault injection campaign is approximately as long as the execution of the software tested.
- **Initialize MFI:** The initialization of the MFI requires only a few microseconds (5–30 s). The values that must be transmitted to the MFI include the following:
  – What attack should be performed (in this case, stuck-at-one).
  – When the attacked should occur.
  – What memory address should be attacked.
- **Run the Software with Fault Injection:** After the MFI is initialized, the execution of the software is started. When the predefined time is reached, the MFI starts the fault injection. All of the software outputs are stored. If the execution of the software requires more than twice the execution time of the golden model, the system is stopped.
- **Dump the Memory:** To determine whether the attack was successful, the memory of the whole system is dumped out of the device to the PC.
- **Evaluate Results:** The evaluation step checks what effect the attack had on the system. The effects can include the following:
  – Fault injected into memory.
  – Output changed.
  – Output and memory content changed.
- **Report:** This step writes the information collected during this test to a file.

Finally, the results of the RAM fault emulation are shown. Table 8 shows that we used a total of 157 k fault injection runs. The execution of the whole test requires 22 h. In 122 k attack scenarios, the fault injection had no effect because the memory cell we attacked was never used after the attack. 35 thousand attacks changed the content of the RAM, but they had no effect on the non-volatile memory or the output of the system. In the rest of the attack cases, the fault detection algorithms of the smart card operating system detects the fault injection. There was no attack case where the system was successfully attacked. The system never sent incorrect outputs or changed non-volatile memory content.

4.9. Comparison to existent fault injector solutions

In recent years, many high-performance fault injection emulation platforms have been published. While fault injection speed is only one parameter to quantify the performance of a fault injector solution, it is important to measure how many faults can be injected in a reasonable time. It is especially important for complex system-on-chip designs with a high number of interesting injection points. In [6], fault injection campaigns using different benchmark configurations have been compared to typical simulation solutions. To cover all design approaches considered, reconfiguration techniques are also presented in [12]. This publication shows two implementations, one using autonomous emulation (AE) and one using partial reconfiguration (PR). The test object in this case is the freely available CORDIC16 core.

Our fault injector solution profits from a higher clock frequency and state-of-the-art FPGA hardware. Another difference that must be considered is that, because of the pattern approach, the fault injection rate is not constant. Therefore, the total number of activated faults is calculated based on the number of activated faults per turn. For the evaluation, the worst case is assumed, where only one fault is injected per pattern. If the simultaneously injected faults are increased, the number of faults per seconds will be increased (one turn consists of all possible bit combinations of the fault pattern width selected). The results of these investigations are compared to our result in Table 9. Because of the different test systems the values have to be normalized. The last column shows the fault injection speed normalized to a clock cycle of 25 ns.

The results of this work presented in Table 9 shows the maximum speed that the fault pattern can be changed per second. This number is important if an attack is emulated by a laser moving over the chip. Thus, it is possible to change the fault effect in the range of 300 k–2 M times per second. If one fault per run should be injected, the whole test requires the run time of the program that should be tested in addition to the configuration time of the MFI.

The comparison shows that the in-system solution does not suffer from additional performance penalties. Strong spatial containment of the possible fault locations leads to very high injection results if combined with large fault injection patterns. It can also be concluded that autonomous emulation provides injection speed advantages compared to partial reconfiguration techniques. The reconfiguration approach suffers not only from limiting communication interfaces but also from long delays caused by the reconfiguration process. This reconfiguration also does not include the time needed to generate reconfigurable sub-blocks. These problems are targeted by the work presented in [14]. In this publication, partial reconfiguration is compared to direct reconfiguration using an FPGA internal port to its configuration memory. While the reconfiguration speed is greatly improved, it is still slower than autonomous emulation solutions such as the one proposed in this paper.

5. Conclusion

This paper presents a case study on multiple fault dependability and security evaluation. A highly modularized controller solution for portable fault injection systems was used. It has been shown that fault injection campaigns can be executed very efficiently through a generalized interface using a high level abstraction of physical fault sources. The design is scalable to allow both fully automated campaigns with a larger fault pattern memory and more user controlled campaigns using a small silicon footprint. The performance is comparable to existing platforms using circuit manipulation and significantly faster than ones using partial and complete FPGA reconfiguration. This study is the first step towards a complete fault injection platform for dependability and security evaluations. Its flexible design will allow the evaluation engineer to shift focus from complex testing architectures to more complex fault models, which should finally allow the modeling of not only simple SEUs but also complex fault attack scenarios. Through the additional abstraction level, a better separation of the test and design engineering tasks is provided. The knowledge gained from these experiments will be used to obtain a better understanding of inner-chip fault mechanics. Such full scale investigations require fully automated saboteur injection techniques, which are currently under development. Consideration of such attacks will be necessary to design efficient and secure smart card systems and is also valid for highly integrated systems or systems under high environmental stress.

Acknowledgments

The authors thank the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749. We would also like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

References

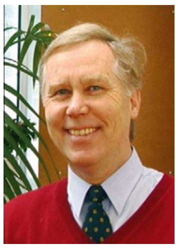
[1] M. Nicolaidis, Time redundancy based soft-error tolerance to rescue nanometer technologies, in: Proc. IEEE VLSI Test Symposium (1999), 1999, pp. 86–94. doi: 10.1109/VTEST.1999.766651.
[2] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, C. Whelan, The sorcerer's apprentice guide to fault attacks, Proceedings of the IEEE 94 (2) (2006) 370–382, http://dx.doi.org/10.1109/JPROC.2005.862424.
[3] D.D. Andres, J.C. Ruiz, D. Gil, P. Gil, Fades: a fault emulation tool for fast dependability assessment, in: Proc. IEEE Int. Conf. Field Programmable Technology (FPT 2006), 2006, pp. 221–228. doi: 10.1109/FPT.2006. 270315.
[4] Xilinx, fpga family. <http://www.xilinx.com/products/virtex5/index.htm> (07.10).
[5] J. Boue, P. Petillon, Y. Crouzet, Mefisto-l: a vhdl-based fault injection tool for the experimental assessment of fault tolerance, in: 28th Annual International Symposium on Fault-Tolerant Computing, 1998. Digest of Papers, 1998, pp. 168–173. doi: 10.1109/FTCS.1998.689467.
[6] P. Civera, L. Macchiarulo, M. Rebaudengo, M.S. Reorda, M. Violante, An fpga-based approach for speeding-up fault injection campaigns on safety-critical circuits, Journal of Electronic Testing 18 (3) (2002) 261–271, http://dx.doi.org/10.1023/A:1015079004512.
[7] A. Benso, B. Prinetto, Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation, Kluwer Academic Publishers, 2003.
[8] P. Ellervee, J. Raik, K. Tammemae, R.-J. Ubar, Fpga-based fault emulation of synchronous sequential circuits, IET Computers & Digital Techniques 1 (2) (2007) 70–76, http://dx.doi.org/10.1049/iet-cdt:20050065.
[9] Gaisler, Leon3 processor. <http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53> (07.10).
[10] A. Pellegrini, K. Constantinides, D. Zhang, S. Sudhakar, V. Bertacco, T. Austin, Crashtest: a fast high-fidelity fpga-based resiliency analysis framework, in: Proc. IEEE Int. Conf. Computer Design (ICCD 2008), 2008, pp. 363–370. doi: 10.1109/ICCD.2008.4751886.
[11] L. Antoni, R. Leveugle, M. Feher, Using run-time reconfiguration for fault injection in hardware prototypes, in: Proc IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT 2002), 2002, pp. 245–253. doi: 10.1109/DFTVS.2002.1173521.
[12] C. Lopez-Ongil, L. Entrena, M. Garcia-Valderas, M. Portela, M.A. Aguirre, J. Tombs, V. Baena, F. Munoz, A unified environment for fault injection at any design level based on emulation, IEEE Transactions on Nuclear Science 54 (4) (2007) 946–950, http://dx.doi.org/10.1109/TNS.2007.904078.
[13] M. Jeitler, M. Delvai, S. Reichor, Fuse – a hardware accelerated hdl fault injection tool, in: Proc. SPL Programmable Logic 5th Southern Conf., 2009, pp. 89–94. doi: 10.1109/SPL.2009.4914906.
[14] L. Kafka, Analysis of applicability of partial runtime reconfiguration in fault emulator in xilinx fpgas, in: DDECS '08: Proceedings of the 2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, IEEE Computer Society, Washington, DC, USA, 2008, pp. 1–4. doi: 10.1109/DDECS.2008.4538781.
[15] B.F. Dutton, M. Ali, C.E. Stroud, J. Sunwoo, Embedded processor based fault injection and seu emulation for fpgas, in: Proc International Conf. on Embedded Systems and Applications 2009, 2009, pp. 183–189.
[16] L. Kafka, M. Danek, O. Novak, A novel emulation technique that preserves circuit structure and timing, in: International Symposium on System-on-Chip 2007, 2007, pp. 1–4. doi: 10.1109/ISSOC.2007.4427437.
[17] R. Leveugle, Early analysis of fault-based attack effects in secure circuits, IEEE Transactions on Computers 56 (10) (2007) 1431–1434, http://dx.doi.org/10.1109/TC.2007.1078.
[18] R. Leveugle, K. Hadjiat, Multi-level fault injections in vhdl descriptions: alternative approaches and experiments, Journal of Electronic Testing 19 (5) (2003) 559–575, http://dx.doi.org/10.1023/A:1025178014797.
[19] J.-M. Daveau, A. Blampey, G. Gasiot, J. Bulone, P. Roche, An industrial fault injection platform for soft-error dependability analysis and hardening of complex system-on-a-chip, in: Reliability Physics Symposium, 2009 IEEE International, 2009, pp. 212–220. doi: 10.1109/IRPS.2009.5173 253.
[20] J. Baraza, J. Gracia, D. Gil, P. Gil, Improvement of fault injection techniques based on vhdl code modification, in: Tenth IEEE International High-Level Design Validation and Test Workshop 2005, 2005, pp. 19–26. doi: 10.1109/HLDVT.2005.1568808.
[21] J.-C. Baraza, J. Gracia, S. Blanc, D. Gil, P.-J. Gil, Enhancement of fault injection techniques based on the modification of vhdl code, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16 (6) (2008) 693–706, http://dx.doi.org/10.1109/TVLSI.2008.2000254.
[22] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock, J. Haid, Automatic saboteur placement for emulation-based multi-bit fault injection, in: Proc. 6th Int Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC) Workshop, 2011, pp. 1–8. doi: 10.1109/ReCoSoC.2011.5981521.
[23] C. Pohl, R. Fuest, M. Porrmann, vMAGIC – automatic code generation for VHDL, Newsletter Edacentrum 2 (2010) 7–10.
[24] C. Bachmann, A. Genser, C. Steger, R. Weiss, J. Haid, Automated power characterization for run-time power emulation of soc designs, in: Proc. 13th Euromicro Conf. Digital System Design: Architectures, Methods and Tools (DSD), 2010, pp. 587–594. doi: 10.1109/DSD.2010.38.

[25] A. Pellegrini, V. Bertacco, T. Austin, Fault-based attack of RSA authentication, in: Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE), 2010, pp. 855–860.
[26] M. Neve, E. Peeters, D. Samyde, J.-J. Quisquater, Memories: a survey of their secure uses in smart cards, in: Proc. Second IEEE Int. Security in Storage Workshop SISW '03, 2003. http://dx.doi.org/10.1109/SISW.2003.10004.
[27] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock, J. Haid, Modular fault injector for multiple fault dependability and security evaluations, Digital System Design (DSD), 2011 14th Euromicro Conference, Aug. 31 2011–Sept. 2 2011, pp. 550–557, http://dx.doi.org/10.1109/DSD.2011.76.

Johannes Grinschgl received his Master's degree in Telematics from Graz University of Technology in 2008, focusing on microelectronics and telecommunications. Since 2010 he is doing his Ph.D. in Electrical Engineering at the Institute for Technical informatics at Graz University of Technology in collaboration with Infineon Technologies Austria AG and Austria Card GmbH. His research interests incorporate fault emulation as well as fault modeling.

Armin Krieg received his Master's degree in Telematics from Graz University of Technology in 2008, focusing on microelectronics and system-on-chip design. Since 2010 he is doing his Ph.D. in Electrical Engineering at the Institute for Technical Informatics at Graz University of Technology in collaboration with Infineon Technologies Austria AG and Austria Card GmbH. His research interests incorporate fault emulation as well as fault detection and recovery.

Christian Steger received the Dipl.-Ing. degree (equivalent to the American Master of Science) 1990, and the Dr. techn. degree in electrical engineering from Graz University of Technology, Austria, in 1995, respectively. He graduated from Export, International Management and Marketing course in June 1993 at Karl-Franzens-University of Graz. In January 2010 he completed the Entrepreneurship Development Program at MIT SLOAN SCHOOL OF MANAGEMENT in Boston, USA. He is key researcher at the Virtual Vehicle Competence Center (ViF, COMET K2) in Graz, Austria. From 1989 to 1991 he was software trainer and consultant at SPC Computer Training Ges.m.b.H., Vienna. From 1990 to 1991 he was research engineer at the Institute for Technical Informatics, Graz University of Technology, Austria. Since 1992 he is Assistent Professor at the Institut for Technical Informatics, Graz University of Technology. He has joined the Reactive Systems Group at Saarland University as interim professor (Lehrstuhlvertreter) in the winter semester 2010/2011. He heads the HW/SW codesign group (8 Ph.D. students) at the Institute for Technical Informatics. His research interests include embedded systems, HW/SW codesign, HW/SW coverfication, SOC, power awareness, smart cards, UHF RFID systems, multi-DSPs. He is member of the IEEE and member of the OVE (Austrian Electrotechnical Association).



Reinhold Weiss is a Professor of Electrical Engineering (Technical Informatics) and head of the Institute for Technical Informatics at Graz University of Technology, Austria. He received the Dipl.-Ing. degree, the Dr.-Ing. degree (both in Electrical Engineering) and the Dr.-Ing.habil. degree (in Realtime Systems) from the Technical University of Munich in 1968, 1972 and 1979, respectively. In 1981, he was as a Visiting Scientist with IBM Research Laboratories in San Jose, California. From 1982 to 1986 he was Professor of Computer Engineering at the University of Paderborn (Germany). He is author and co-author of about 170 scientific and technical publications in Computer Engineering. His research interests focus on Embedded Distributed Real Time Architectures (parallel systems, distributed fault tolerant systems, wearable and pervasive computing). He is a member of the International Editorial Board of the Computers and Applications (ISCAs) journal. Further, he is a member of IEEE, ACM, GI (Gesellschaft fuer Informatik, Germany), and OVE (Austrian Electrotechnical Association).

Holger Bock received his Diplom Ingenieur (corresponding to Master) degree in electrical engineering at the Graz University of Technology in 1994. From 1991 to 1998 he has been working on concepts, software and hardware development, especially on VLSI-Design for cryptographic coprocessors for smart cards (DES, ECC) at the Institute for Applied Information Processing and Communications Technologies (IAIK). In December 1998 he joined the team at Infineon's development center in Graz as a core competence for security. Since beginning of 2001 he had been a member of the technology and innovations methodology team at Infineon's business group Chipcard and Security ICs, focussing on secure, especially DPA resistant, design methodologies for cryptographic hardware. In October 2006 he has become responsible for worldwide funding management for Infineon's business group Chipcard.

Josef Haid received a Master's degree in Telematics and a doctoral degree in electrical engineering both from the Technical University of Graz in Austria in the years 2001 and 2003 respectively. Presently he is a senior staff engineer at Infineon Technologies in Graz/Austria and is responsible for specification of low-power contactless smart cards. His interests include advanced digital design and low power design of hardware and software.

Thomas Aichinger received his Diplom Ingenieur (corresponding to Master) degree in computer science at the Vienna University of Technology in 2003. From 2002 to 2007 he has been working on assessments of smartcards and software used for qualified electronic signatures, as well as an assessor for electronic payment systems and for data protection at the A-SIT (Secure Information Technology Center – Austria). In June 2007 he joined the R&D department at Austria Card Plastikkarten und Ausweissysteme GmbH responsible for security certification (Common Criteria, CAST, VISA Risk), definition of security requirements and concepts for their implementation in the ACOS Smart Card Operating System. Since 2011 he is also Product Manager for ID Products.

Christiane Ulbricht received her Diploma (M.Sc.) in October 2002. After that she worked as a research assistant at the Institute of Computer Graphics and Algorithms at the Vienna University of Technology. Her Ph.D. thesis focused on photo realistic computer graphics and especially on verifying the correctness of physically based computer graphic systems. In December 2006, she joined Austria Card as project manager at the research and development department. She was promoted to head of project management office in June 2008. Since June 2011, she has become head of the Research and Development department at Austria Card.

A Side Channel Attack Countermeasure using System-On-Chip Power Profile Scrambling

Armin Krieg, Johannes Grinschgl,
Christian Steger and Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology
Graz, Austria
{armin.krieg, johannes.grinschgl,
steger, rweiss}@tugraz.at

Josef Haid
Design Center Graz
Infineon Technologies Austria AG
Graz, Austria
josef.haid@infineon.com

***Abstract*—Since the discovery that hardware used for cryptographic applications could leak secret information through its power or radiation profile a wide range of possible attack methods has been published. The rapid evolution of these side-channel attacks made it increasingly important to minimize this possible information leakage. Additionally timing information also derived from this power profile is used to control fault-attack campaigns to drive the system into an unintended state. Therefore a wide range of leakage countermeasures has been developed for dedicated cryptographic hardware. Contrariwise only little work is available concerning power profile scrambling techniques for cryptographic software implementations running on general purpose architectures. Such implementations often include power management hardware to cope with several power budget constraints which could be used to influence the system's power consumption during run-time. This paper proposes a novel side channel attack countermeasure technique using such power management methods in combination with techniques for power profile manipulation. State-of-the-art power estimation hardware using a reduced power model allows for the efficient on-line monitoring and manipulation of the power consumption and radiation profile.**

I. Introduction

Since the publication of power analysis attacks as a form of side channel attacks numerous countermeasures have been presented to close this dangerous information channel. The focus of most of these publications concerns the improvement of cryptographic hardware implementations on different abstraction levels. These measures are often not applicable to general purpose architectures because they rely on specialized hardware or measures that need development effort that may not be wanted for specific low-effort projects. For such low-cost applications generally general-purpose architectures are used which mostly do not include any security-aware hardware optimizations. The design challenges of such architectures for security applications are summarized in [1].

Most hardware countermeasures described in literature concern specialized logic styles and design flows. For the designer of a low-cost embedded system this poses several problems. First, specialized logic styles often lead to significantly increased production costs [2]. Second, circuit-style changing countermeasures cannot be improved after production if they are proven to be insecure. Such problematic combinations of masking and dual-rail logic have been investigated in [3]. Other approaches using circuitry to produce artificial power consumption are problematic in systems with a limited power budget.

Because of the ever increasing integration density for system-on-chips numerous publications presented power-aware circuit-design and energy management approaches. These power estimation and management circuits could be used for a targeted approach to counteract treacherous power profile variations independent from the used cryptographic algorithm.

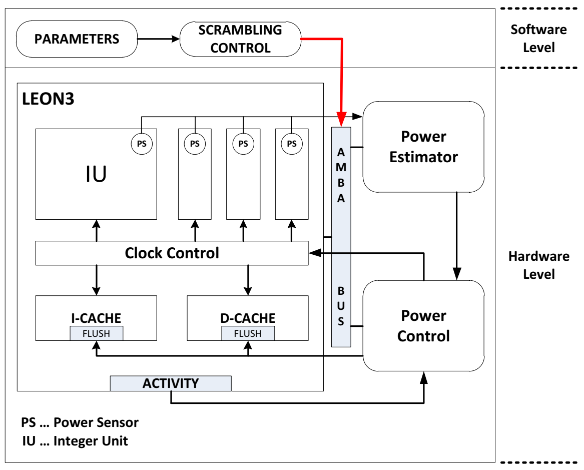


Fig. 1. Power control and scrambling architecture

Therefore we propose the use of specialized power emulation hardware in combination with power management techniques to reduce or scramble analyzable power consumption profile peaks. A basic overview over our proposed system architecture is given in Figure 1. Our approach will be adaptable during run-time to provide protection for different algorithms and to change its behavior if flaws are detected after market entry. The resulting behavior of an augmented system will be analyzed using our in-house security evaluation platform. The main contributions of this work are:

- An extended power management approach using hardware-accelerated power estimation techniques.
- The introduction of a power manipulation approach to improve the system's side channel and fault attack robustness characteristics.
- A case study using a commonly used software implementation of the AES algorithm.

This paper is structured as follows. Section II briefly reviews the state-of-the-art concerning side channel attack hardware countermeasures for general purpose architectures. Followed by Section III showing power analysis methods to evaluate the effectiveness of our implementation. In Section IV technical details of our chosen hardware-accelerated power estimation approach and its application are provided. Section V presents our power manipulation concept and its advantages to the state-of-the-art. Experimental results using our simulation and evaluation platform are presented in Section VI. Finally our results are concluded and some details about our future work are given in Section VII.

II. Related Work

For this work we will not consider countermeasures that are not applicable to general purpose architectures. Such techniques would include specialized logic styles and approaches that significantly change the way an architecture works as these solutions would not be used for such general purpose systems (such as [4]).

The most research in the field of power analysis countermeasures concerned the flattening of the power profile using direct methods. One technique to reduce power variations is by generating artificial energy consumption like shown in [5]. While being very efficient this method increases the power demand of the system by up to 70 percent. A similar approach has been presented by Ratanpal et al. in [6].

Ambrose et al. introduced several new approaches to reduce power variations by instruction- and architecture level techniques. In [7] they proposed such a method based on random code injection to randomize the power consumption of the system. While the areal effort of this approach is very low it results in large runtime and energy overheads of up to over 60 percent depending on the used cryptographic algorithm. Specifically targeting the AES algorithm they also proposed a multi-processor power balancing technique in [8]. This work reduces runtime-overhead to nearly zero but requires a second identical processor core to process the same instructions as the first with inverted data as an operand. While this core can be used for normal operation if no encryption takes place, it still forms a significant silicon area overhead.

Another generic approach for cryptographic hardware implementations has been presented by Tiri et al. [2]. Their solution to reduce data-dependent power variations is a secure digital design flow using a secure logic style (wave dynamic differential logic - WDDL). They applied this methodology on a hardware implementation of the AES algorithm. According to their results this had a tremendously negative impact on the gate count and maximum achievable frequency.

An energy-aware approach using clock-gating hardware has been presented in [9], but the authors only used randomization to achieve power masking. Their implementation neither uses feedback from a power-estimation source nor is it adaptable during run-time. The approach presented in [10] extends the formerly shown solutions using existing dynamic voltage and frequency switching hardware. The authors also show that naive power randomization techniques that are applied every clock cycle are not sufficient to counteract power analysis attacks.

Most of these countermeasures do not concern general purpose processors for low-cost hardware implementations. They either imply extensive changes to the hardware structure or decrease area efficiency by a significant factor. Therefore we introduce a modular approach using power estimation hardware that can also be used for power management purposes. All hardware modeling code modifications are done in an automatized way and additional controller hardware is small enough to be added to even low-cost systems. The scrambled power consumption profile makes it also more difficult to find suitable time points for fault attacks.

III. Power Analysis (PA)

Power analysis utilizes the fact that the power consumption and emitted radiation of a given system does not only depend on the executed code sequences but also on the processed data. This unintentionally supplied information can be used to gain timing and statistical knowledge about the executed algorithm and keys. Therefore this method has become very popular with cryptologists and adversaries alike in recent years.

A. Simple Power Analysis (SPA)

SPA denotes a specific form of power analysis in which the power consumption or radiation profile of a cryptographic system is analyzed in a direct manner. This means that a low number of traces is examined for specific characteristics of the implementation. Such characteristics could be timing information or more high level data like clearly visible execution branches. It is therefore necessary to know at least basic characteristics of the used algorithm and its possible software or hardware implementations.

This direct power profile interpretation can also be used as a base for differential fault analysis as shown in [11]. In this case an intentional fault is injected into a cryptographic system to disturb the execution in a defined way. Information provided by SPA can be of vital importance for a successful fault analysis attack.

Countermeasures: As SPA relies mainly on the direct interpretation of the extracted power or radiation data, it is sufficient to reduce the dependency of these profiles from the data that is processed. This can be done by profile scrambling or execution randomization.

B. Differential Power Analysis (DPA)

As for SPA it is of importance to know the underlying cryptographic algorithm, research focused on statistical methods to overcome this complication. Therefore DPA as an advanced form of power analysis has been published by Kocher et al. in 1999 [12]. Because it relies on the statistical evaluation of a high amount of power consumption traces it has been proven to be a powerful tool to extract secret key information from cryptographic systems. This technique exploits the fact that the power profile or the radiated electro-magnetic field is dependent on how many logic gates are switching at the same time.

Therefore there also exists a relationship between the form of the power or radiation profile and the state of the used key bits. The techniques originally presented in [12] still required a high amount of power traces to successfully attack a system if even basic countermeasures are available. Therefore recent years showed a variety of new high performance approaches using correlation techniques such as those shown in [13] and [14].

Countermeasures: This leakage is caused by asymmetries of standard CMOS gates and therefore more sophisticated countermeasures are needed than against SPA attacks. It is of vital interest to the designer of a cryptographic system to make statistical analysis as difficult as possible. In the optimum case the dependency of processed data and execution profile is broken. Current countermeasures include masking or randomization techniques as well as approaches that reduce power profile variations directly using specialized logic styles.

IV. Hardware-accelerated Power Estimation (PE)

The power emulation methodology has been introduced as a promising alternative to simulation-based power profiling approaches. Its hardware-accelerated nature makes its possible to implement power estimation hardware into a system to gain intermediate knowledge about its power consumption for further processing.

The principle of *power emulation* (PE), as initially introduced in [15], is based on state-of-the-art functional emulation that is being augmented with hardware-implemented power models. Thereby, the power estimation process and the functional emulation are executed concurrently during the run-time of the system. Contrary to the original design used for power evaluation approaches the power estimation data can be directly used for power management purposes.

A. Power Estimation Unit

This work is based on the high-level *power estimation unit* as initially introduced in [16]. The unit utilizes additive linear power macro models as expressed in Equation 1 to generate static and dynamic power consumption estimates for different system components by monitoring power-relevant signals x_i. The basic working principle of the proposed power emulation and manipulation architecture is depicted in Figure 2.

$$\hat{P}[t] = \hat{P}_{sta} + \hat{P}_{dyn}[t] = c_0 + \sum_{i=1}^{N} c_i x_i[t] \qquad (1)$$

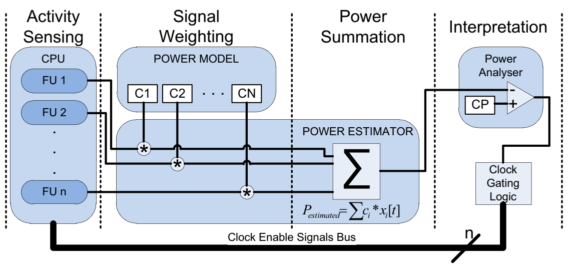


Fig. 2. Power estimation and manipulation principle

To remove or scramble power profile variations these have to be detected at the earliest possible moment. Otherwise it is not possible to counteract peaks before they start to manifest themselves in a detectable manner. Also for power management purposes the high accuracy of a detailed power emulation unit is not necessary. For integration into in-system power management solutions it is sufficient to detect strong power variations, the accurate amount of power currently consumed is of no immediate concern. Therefore the required circuit-size can be reduced to a state just providing an acceptable level of accuracy.

The automatized characterization process presented in [17] is used to detect inter-signal correlation to reduce redundancy. This flow also provides verification tools to determine the grade of estimation accuracy by comparison to gate-level simulations done for a given set of benchmark applications.

B. Power Characterization

A reduced power model containing only state-dependent model coefficients has been generated using the power characterization methodology shown in [17]. The power characterization flow can be schematically described using the following stages:

1) **Generation of training-set data**: A set of microbenchmarking applications, covering a wide range of different application scenarios is simulated using state-of-the-art gate-level power estimation tools. The gate-level description of our system-under-evaluation is derived using synthesis tools provided by our industry partner. This simulations result in a broad collection of signal activity as well as power estimation data.
2) **Parameter selection**: From this enormous amount of training-set data it is now of interest to determine power-relevant signals, i.e., power model parameters x_i. This selection process has been automatized as presented in [17].
3) **Coefficient fitting**: The parameters chosen in the previous stage are now used in a model coefficients fitting process to determine coefficients c_i. This fitting process is based on a non-negative linear regression technique.

V. PA Countermeasure Concept

The following power analysis countermeasure concept is based on three part approach:

1) **Power Estimation**: State-of-the-art power emulation hardware is used to approximate the present power consumption with a delay of only a few clock cycles. To guarantee earliest possible reaction to power consumption variations only signals from the decode stage of the processor pipeline have been selected for the characterization process as shown in Figure 3.
2) **Power Profile Manipulation**: To reduce the currently needed power budget clock gating is applied on a global level, but the approach is flexible enough to be changed to a more fine grained gating method. Artificial consumption is generated by code independent cache flushing, but also other system-internal modules could be triggered to gain a similar effect. As the estimation result always directly depends on the executed code, the manipulation quality can be improved by adding a true random element to this simple calculation.
3) **Power Control**: A power distribution module has been implemented as a central point to gate or distort the current power consumption profile depending on the current estimation result. The support of module-level clock gating and power generation is assured by a small monitoring module to keep track of currently active and idle system elements.

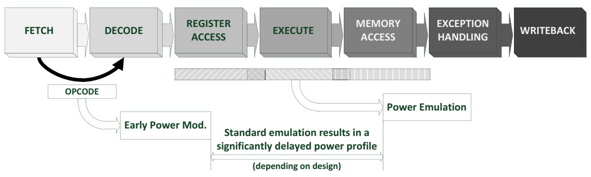


Fig. 3. Reduction of Power Estimation Delay

To keep hardware requirements as small as possible the power distribution module is implemented as a simple programmable lookup-table like control unit. Like the power estimation unit it is also designed to keep decision latency as low as possible. A basic overview over our proposed power scrambling architecture is shown in Figure 1. The following aspects can be adjusted to achieve an optimized power scrambling performance depending on the chosen application:

- Running average sample length (min. 2 samples)
- Clock-gating look-up-table (LUT)
- Power manipulation LUT

The run-time power manipulation flow is schematically depicted in Figure 4. As shown in this figure the power estimation result is first differentiated and averaged (2-sample). This way the hardware is able to detect strong positive or negative power consumption deviations. The result of this process is then further processed to determine proper actions.

The power manipulation logic first applies a user definable running average filter on positive input values. Negative values are not considered and therefore a longer averaging process results in less strong variations. This way the sensitivity of the power profile scrambler can be increased significantly. This design decision to include only positive values leads to a significantly smaller controller logic while the user looses only few influence on the decision result.

Additionally a small pre-processing average filter can be activated to influences how fast the power manipulation unit reacts to power peaks. This value is especially important as the counteraction performance strongly depends on its reaction speed [10]. If strong positive variations are detected a clock gating strategy will be applied according the user programmed gating LUT. The user has the ability to set the length this clock gating technique should be applied. In case of strong negative power variations cache flushing is triggered as long as defined by the according LUT.

The LUT parameters have to be chosen very carefully otherwise high clock-gating and cache flushing rates will have a very high negative impact on the system's performance. These LUT and averaging filter parameters can be easily set using special function registers provided through the AMBA interface of the power manipulation module.

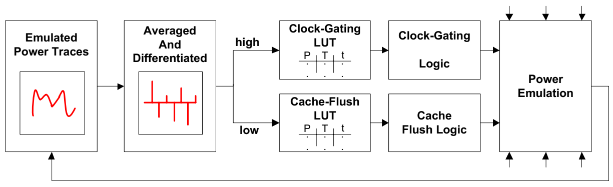


Fig. 4. Continuous run-time power manipulation flow

A. Advantages compared to state-of-the-art

Existing power profile scrambling solutions based on power-management techniques as presented in Section II are either only relying on random number generators or need dynamic voltage and frequency scaling (DVFS) capable hardware. Previous research has shown that frequency scaling methods could be ineffective if advanced power analysis techniques are applied [18]. The advantages of our chosen approach can be concluded in the following way:

1) **Reconfigurability**: The power profile scrambling mechanism can be configured during runtime. This allows for optimum support of different cryptographic algorithms and to adapt existing mechanisms if weaknesses have been detected. Scrambling parameters can be adjusted according to the currently executed workload to improve performance for non-security critical code sections.
2) **Low-Cost Solution**: No specialized logic style or routing strategies are needed. Existing run-time power monitoring and management circuitry can be partially reused to implement security functionality. No additional hardware like random number generators or current sinks is needed.
3) **Extendability**: If existing random number generation hardware is available, our approach can be easily extended to include additional power profile randomization. Also it can be easily extended to provide a finer grained clock-gating method and to add additional targets to generate non-execution dependent power consumption.

VI. Experimental Results

To prove the usability of our approach we chose a widely used general purpose processor and AES software implementation. The hardware platform consists of an open source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler [19] called LEON3. For the AES implementation we chose the widely known C implementation from [20]. The system-under-test has been synthesized using Xilinx ISE software and tested using the ML507 evaluation board from the same manufacturer. All micro-benchmark characterization processes have been done using gate-level power simulations provided through Synopsys PrimeTime made available by our industrial partner. For these simulations a production quality 90nm standard cell library has been chosen. The proposed power estimation hardware then emulates the behavior of a system synthesized using this library. The FPGA-hardware is only used for the emulation of the system including its hardware-accelerated power model. The chosen configuration of the LEON3 processor is summarized in Table I. LEON3 and the Xilinx Virtex5 FPGA-platform have only be chosen for exemplary tests, all proposed hardware elements provide general interfaces to guarantee platform independence (except clock and cache interfaces which have to be adjusted accordingly). All external analyses have been done using MathWorks Matlab 2009b on a six-core 3.2 GHz AMD Phenom-II machine with eight gigabytes of RAM.

TABLE I
LEON3 Processor Configuration

Operating Frequency [MHz]	40
Instruction Cache Sets	2
Instruction Cache Set Size [kB]	4
Data Cache Sets	1
Data Cache Set Size [kB]	4
MMU TLB Entries	8
MMU Page Size [kB]	4

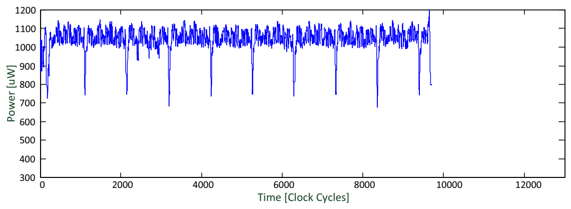
For our chosen decode-only power emulation approach a power model containing only ten coefficients has been generated. This model has been verified using power simulation data and an average relative error of lower than 15 percent could be determined for our micro-benchmark applications. Again we have to mention that a rough power estimation accuracy is sufficient if only strong variations should be detected.

Based on balanced standard settings of the Xilinx XST synthesis tools for the Virtex5FXT FPGA the LEON3 processor including all additional modules has been synthesized. The synthesis results are concluded in Table II. Additionally the system overhead (OH) is given relatively to the complete system including all peripherals.

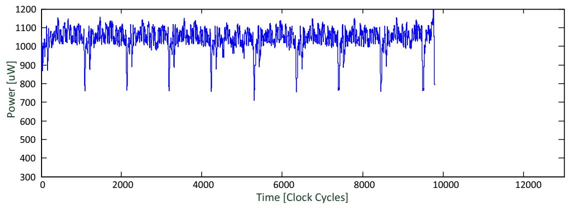
TABLE II
Synthesis Results - Virtex5 - Balanced Strategy

Unit	Slice Registers	OH [%]	LUTs	OH [%]
Complete System	8213	-	15998	-
LEON3 Core	3509	42.7	7271	45.4
Power Distributor Logic	88	1.07	157	0.98
Power Estimation Logic	100	1.22	234	1.46

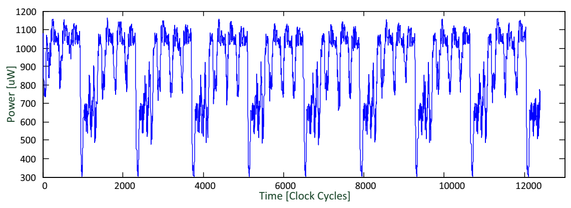
The complete system has been simulated using the ModelSIM RTL simulation software. An unmodified version of our chosen AES software implementation has been simulated using different parameters for the power manipulation unit. Figure 5a shows the trace of the AES computation without any activated power manipulation. All ten AES rounds of the chosen implementation can be clearly seen in this power trace. An unprotected software implementation therefore results in a significantly amount of timing and switching information leaking through its power profile.



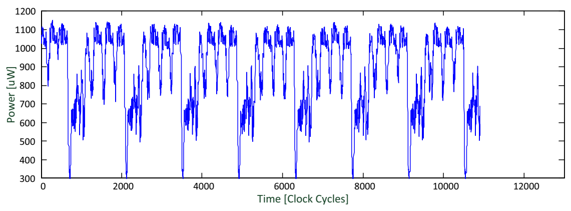
(a) Single AES-Trace without any power manipulation



(b) Single AES-Trace using a light clock-gating strategy



(c) Single AES-Trace using power dependent cache flushing



(d) Single AES-Trace using light clock-gating and cache flushing

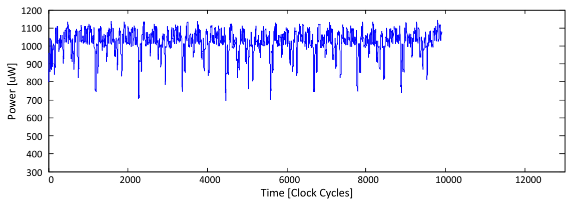
Fig. 5. Tracing and DPA analysis results without internal averaging

In Figure 5b a single AES trace is shown while a power dependent clock gating strategy is activated. It can be seen that a significant amount of additional noise is added to the power trace while execution performance is only slightly influenced.

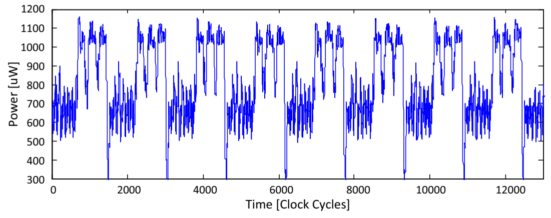
Figure 5c shows a single AES trace manipulated by a power dependent cache flushing strategy. While this strategy increases noise level substantially more than a pure clock gating method, it also has a much higher impact on operating performance.

A combined approach using both clock-gating and cache-flushing methods is depicted in Figure 5d. This solution combines the advantage of very high noise levels with a decreased power consumption of the whole system. Because of the time intensive nature of cache flushes the execution performance is influenced negatively.

Figure 6a shows a single AES trace for a pure clock-gating configuration using an internal 16-sample running average filter. It can be seen that this additional filter provides a better internal



(a) Single AES-Trace using light clock-gating strategy



(b) Single AES-Trace using power dependent cache flushing

Fig. 6. Tracing and DPA analysis results using 16-sample averaging

distribution of profile variation points and therefore leads to more generated noise because of a wider manipulation point distribution.

In Figure 6b the same averaging filter configuration has been used in combination with the cache flushing technique. The performance of the system is further decreased but also the power consumption has been lowered.

TABLE III
POWER MANIPULATION INFLUENCE ON OPERATING PERFORMANCE - AVERAGED OVER 1000 TRACES EACH

Strategy	Power [uW]	OH [%]	Time [Cycles]	OH [%]
No Management	1029	-	9700	-
Software Countermeasure	1045	1.55	9700	-
Clock-gating (no averaging)	1031	0.19	9800	1
Cache-flushing (no averaging)	884	-14.1	12400	27.8
Combined (no averaging)	883	-14.2	10900	12.4
Clock-gating (16 Samples)	1009	-1.9	10000	3
Cache-flushing (16 Samples)	820	-20.3	14000	44.3

Table III gives a short overview over the impact of the chosen power manipulation techniques on the operational performance of this system. These average values describe the mean power consumption and execution time of one 128-bit AES encryption. It can be clearly seen that manipulations using cache flushes have a strong impact on execution time. For comparison reasons a low impact software countermeasure has also been implemented (random NOP-instruction injection).

A. DPA attack results

Based on an in-house power emulation simulation and emulation platform DPA attacks are performed using the well known Matlab tools of the OpenSCA project [21]. Power traces are generated using a more detailed power model including data-dependent information as presented in [22].

In Table IV DPA attack results are shown for different configurations of the proposed architecture. It can be clearly seen that while cache-flushing proves to be a simple and effective method to reduce correlation, pure clock gating if applied carelessly proves to

TABLE IV
DPA ATTACK ANALYSIS

Strategy	Corr. Key Bytes of 16	Evaluated Traces
No Management	16	300
Software Countermeasure	16	400
Clock-gating (no averaging)	16	200
Cache-flushing (no averaging)	12	900
Combined (no averaging)	3	900
Clock-gating (16 samples)	2	900
Cache-flushing (16 samples)	0	900

be even counterproductive. The pure clock-gating strategy could be significantly improved using a 16-sample running average filter while operational performance was only worsened by about three percent as shown in Table III. The combination of a cache-flushing technique with a 16-sample averaging filter proved to be most effective strategy for this AES encryption testcase. The pure software countermeasure solution, while not having a significant impact on performance, only provides a small countermeasure effect.

VII. CONCLUSION

This paper presented a novel power analysis countermeasure approach using power manipulation and emulation techniques. This allows for the scrambling of power variations that could be exploited for simple or differential power analysis attacks. Clearly visible timing and power peak relations that could be used for fault-based attacks can also be masked from a possible adversary. Contrary to existing side channel attack hardware countermeasures this approach is applicable to any general purpose architecture providing run-time power information. The flexible approach allows to adapt the method of operation during run-time for protection of different types of cryptographic load. Our approach only slightly increases the system's power consumption and does not rely on non-standard logic styles.

The implemented hardware countermeasures are fully synthesizable and optimized for smallest possible area requirements. Therefore they can be introduced into even very cost constrained cryptographic embedded systems. The feasibility evaluation of the implementation of such countermeasures into low cost smart-card systems will be part of our future work.

ACKNOWLEDGMENTS

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749. We would also like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

REFERENCES

[1] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 461–491, 2004.
[2] K. Tiri and I. Verbauwhede, "A digital design flow for secure integrated circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 7, pp. 1197–1208, 2006.
[3] P. Schaumont and K. Tiri, "Masking and dual-rail logic dont add up," *Cryptographic Hardware and Embedded Systems-CHES 2007*, pp. 95–106, 2007.
[4] P. Grabher, J. Grossschaedl, and D. Page, "Non-deterministic processors: FPGA-based analysis of area, performance and security," in *Proceedings of the 4th Workshop on Embedded Systems Security*. ACM, 2009, pp. 1–10.
[5] R. Muresan and C. Gebotys, "Current flattening in software and hardware for security applications," in *Hardware/Software Codesign and System Synthesis, 2004. CODES+ ISSS 2004. International Conference on*. IEEE, 2005, pp. 218–223.
[6] G. Ratanpal, R. Williams, and T. Blalock, "An on-chip signal suppression countermeasure to power analysis attacks," *IEEE Transactions on Dependable and Secure Computing*, pp. 179–189, 2004.
[7] J. Ambrose, R. Ragel, and S. Parameswaran, "RIJID: random code injection to mask power analysis based side channel attacks," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 489–492.
[8] J. Ambrose, S. Parameswaran, and A. Ignjatovic, "MUTE-AES: a multiprocessor architecture to prevent power analysis based side channel attack of the AES algorithm," in *Proceedings of ICCAD2008*. IEEE Press, 2008, pp. 678–684.
[9] L. Benini, A. Macii, E. Macii, E. Omerbegovic, F. Pro, and M. Poncino, "Energy-aware design techniques for differential power analysis protection," in *Proceedings of the 40th annual Design Automation Conference*. ACM, 2003, pp. 36–41.
[10] S. Yang, W. Wolf, N. Vijaykrishnan, D. Serpanos, and Y. Xie, "Power attack resistant cryptosystem design: a dynamic voltage and frequency switching approach," 2005.
[11] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," *Advances in CryptologyCRYPTO'97*, pp. 513–525, 1997.
[12] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO 99*. Springer, 1999.
[13] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 135–152, 2004.
[14] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Verlag, 2007.
[15] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *Proceedings of the 42nd Annual Design Automation Conference*. ACM, 2005, pp. 700–705.
[16] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss, "An emulation-based real-time power profiling unit for embedded software," in *SAMOS 2009*, 2009, pp. 67–73.
[17] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid, "Automated Power Characterization for Run-Time Power Emulation of SoC Designs," in *DSD 2010*, 2010, pp. 587–594.
[18] K. Baddam and M. Zwolinski, "Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure," in *VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*. IEEE, 2007, pp. 854–862.
[19] Aeroflex Gaisler, "LEON3 Processor," December 2010. [Online]. Available: http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53
[20] B. Gladman, "Implementations of AES (Rjindael) in C/C++ and assembler," December 2010. [Online]. Available: http://gladman.plushost.co.uk/oldsite/cryptography_technology/index.php
[21] E. Oswald, "OpenSCA, An open source toolbox for Matlab," December 2010. [Online]. Available: http://www.cs.bris.ac.uk/home/eoswald/opensca.html
[22] A. Krieg, C. Bachmann, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "Accelerating Early Design Phase Differential Power Analysis Using Power Emulation Techniques," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*, In Press.

POWER-MODES: POWer-EmulatoR- and MOdel-Based DEpendability and Security Evaluations

ARMIN KRIEG, JOHANNES GRINSCHGL, CHRISTIAN STEGER,
and REINHOLD WEISS, Graz University of Technology, Austria
HOLGER BOCK and JOSEF HAID, Infineon Technologies Austria AG

Innovation cycles have been shortening significantly during the last years. This process puts tremendous pressure on designers of embedded systems for security-or reliability-critical applications. Eventual design problems not detected during design time can lead to lost money, confidentiality, or even loss of life in extreme cases. Therefore it is of vital importance to evaluate a new system for its robustness against intentionally and random induced operational faults. Currently this is generally done using extensive simulation runs using gate-level models or direct measurements on the finished silicon product. These approaches either need a significant amount of time and computational power for these simulations or rely on existing product samples.

This article presents a novel system evaluation platform using *power emulation* and *fault injection* techniques to provide an additional tool for developers of embedded systems in security-and reliability-critical fields. Faults are emulated using state-of-the-art fault injection methods and a flexible pattern representation approach. The resulting effects of these faults on the power consumption profile are estimated using state-of-the-art power emulation hardware. A modular system augmentation approach provides emulation flexibility similar to fault simulation implementations. The platform enables the efficient evaluation of new hardware or software implementations of critical security or reliability solutions at an early development phase.

Categories and Subject Descriptors: B.2.3 [**Reliability, Testing, and Fault-Tolerance**]: Arithmetic and Logic Structures

General Terms: Design, Security, Reliability

Additional Key Words and Phrases: Hardware security, hardware reliability, fault attack resistance, power estimation, fault emulation

ACM Reference Format:
Krieg, A., Grinschgl, J., Steger, C., Weiss, R., Bock, H., and Haid, J. 2012. POWER-MODES: POWer-EmulatoR- and MOdel-based DEpendability and Security evaluations. ACM Trans. Reconfig. Technol. Syst. 5, 4, Article 19 (December 2012), 21 pages.
DOI = 10.1145/2392616.2392617 http://doi.acm.org/10.1145/2392616.2392617

1. INTRODUCTION

The increasing silicon integration density brought tremendous challenges for system engineers. First, an ever-increasing number of new features have to be implemented

This work is supported by the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749.
Authors' addresses: A. Krieg (corresponding author), J. Grinschgl, C. Steger, and R. Weiss, Institute for Technical Informatics, Graz University of Technology, Austria; email: armin.krieg@tugraz.at; H. Bock and J. Haid, Infineon Technologies Austria AG, Austria.

DOI 10.1145/2392616.2392617 http://doi.acm.org/10.1145/2392616.2392617

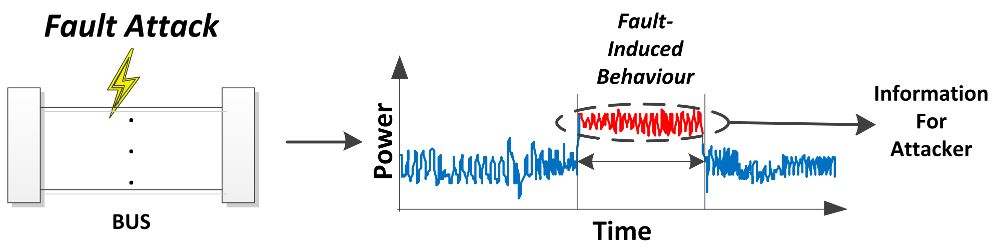


Fig. 1. Influence of security reset on power consumption profile.

in ever-decreasing implementation cycles. Second, a growing number of these highly integrated systems are used in critical fields with high demands on system security and reliability. Therefore system designer support is needed during the design phase to test new hardware and software implementations for possible weaknesses [Ravi et al. 2004].

Especially for high safety applications (e.g., space, automotive, aeroplane) a wide variety of fault injection techniques has been developed [Arlat et al. 1990; Jenn et al. 1994; Leveugle and Hadjiat 2003]. In this field faults are mostly of a random nature and therefore these evaluation approaches rely on a random selection and injection of operational errors. For these platforms generally no fault attacks have been considered because of their controlled nature [Leveugle 2007]. Otherwise for modern security evaluations it is of vital importance to know the impact of faults on the power consumption profile because it forms a direct (Simple Power Analysis SPA) or indirect (Differential Power Analysis DPA) information source to an adversary (e.g., [Kocher et al. 1999; Roche et al. 2011]). To estimate the power profile during runtime hardware accelerated methods like *power emulation* have been proposed. These techniques can be used to close a gap often remaining in classic fault injection platforms for reliability purposes. Such a gap constitutes a serious problem in security-critical systems as even the reset behavior during an attack already provides information to the attacker, as depicted in Figure 1. Such information allows to draw conclusions about internal fault reaction mechanisms like emergency memory accesses that would lead to increased consumption or a decreased power profile if a simple device reboot is triggered (such effects have been shown in Bar-El et al. [2006]).

To evaluate hardware and software implementations for their fault robustness these techniques have been combined to form a fault effect evaluation platform. The main contributions of this work are as follows.

— a novel evaluation flow combining *fault injection*, *power emulation*, *power* and *fault analysis* techniques;
— the introduction of a novel FPGA-based platform for runtime fault-attack and dependability evaluations;
— case studies using a common AES software and hardware implementation and general-purpose processor.

This article is structured as follows. In Section 5 a brief overview over the current state-of-the-art concerning similar evaluation platforms is given. Section 2 contains a short introduction into *fault injection* methods and applications. We follow with Section 3 summarizing the main characteristics of different power analysis methods. Section 4 gives an introduction into power emulation and its application for power analysis purposes. In Section 6 the principle methodology of the fault effect evaluation platform is presented, followed by Section 7 showing some experimental results to prove the viability of our approach. Finally Section 8 gives a short conclusion about our work and some views into future developments in this field.

2. FAULT INJECTION (FI)

The term fault injection describes the intentional introduction of faults into a system to simulate errors that are caused by external influences changing the system's normal behavior. Faults can be injected on several different levels of abstraction.

— *Hardware Level*. This level requires the least amount of system invasion as existing test equipment can be reused. Faults can be injected on pin level by manipulating data and control flow [Arlat et al. 2002]. Internal manipulation is enabled through radiation sources or by direct injection into the silicon device [Gunneflo et al. 2002]. The disadvantage of fault injection on this level is the need for existing prototype samples.
— *Software Level*. The basic principle behind software fault injection is the same as in hardware FI [Tsai et al. 2002]. A comprehensive software testing environment is presented in Segall et al. [2002].
— *Modeling Level*. As shown in Section 5 direct evaluation of hardware description level models is of utter importance to embedded system designers. As shown in Baraza et al. [2002] faults can be introduced using simulator commands or by augmentation of the descriptive code itself.

For this fault injection platform we will concentrate on hardware-description-based approaches. In this case faults can be introduced using simulator commands in hardware model simulators or by adapting the model description itself for hard-accelerated emulation.

2.1. Simulation Based

The advantage of simulation-based approaches is that no changes to the model description itself are necessary. The complete model simulation flow can be reused for fault injection examinations. Faults are activated through simulator commands to manipulate certain states of the system. This can be done because every element of system design is accessible during simulation. The disadvantage of this approach is that a significant amount of computational power is necessary for detailed simulations of complex designs. Especially if the abstraction level is lowered to gate level; lengthy simulation runs are necessary for comprehensive evaluations.

2.2. Adaption Based

Adaption-based fault injection approaches augment the existing hardware description with additional circuitry to influence the behavior of predefined system elements. While specific knowledge about the location of faults manifesting themselves in form of errors is necessary, very high evaluation performances are enabled. Especially in combination with emulation techniques using FPGA prototyping platforms large amounts of different fault patterns can be evaluated in relatively short periods of time [Grinschgl et al. 2011b]. Two different concepts of manipulative elements are generally used.

— *Mutant*. In this case a complete system module is replaced with a manipulated version of this module. During normal operation this module works as designed but after fault activation it behaves like a defective one.
— *Saboteur*. Saboteurs are small elements that are inserted into signal lines. During normal operation these act transparently but after fault activation a predefined fault effect influences the targeted signal.

3. POWER ANALYSIS (PA)

In standard CMOS technology power consumption and emitted electro-magnetical radiation of a given system is dependent on the amount of simultaneously switching transistors. Therefore it not only depends on the control flow of a program but also on the data it processes. This fact is exploited by power analysis approaches using timing analysis or statistical methods to extract secret information like the encryption key through this externally available power information. This powerful tool therefore became a strong companion of adversaries and cryptologists alike.

3.1. Simple Power Analysis (SPA)

In case of SPA, only a small number of power consumption radiation traces are recorded for further analysis. These are analyzed in a direct manner, meaning that these traces are evaluated for specific algorithm- dependent variations and timing characteristics. SPA is therefore based on a variant of control flow analysis of cryptographic workload. To efficiently execute this evaluation at least basic knowledge about the used algorithm or implementation is needed.

Classic programming language elements that result in strongly visible power profile variations are branches. Depending on the branch condition these also lead to strong timing differences that could be detected by an adversary. Therefore software implementations of cryptographic algorithms must never be based on branches to hinder SPA analysis. Another effective countermeasure against such power analysis methods is execution randomization to disconnect execution behavior from algorithmic behavior.

3.2. Differential Power Analysis (DPA)

DPA combines the power or radiation profile analysis techniques presented in Section 3.1 with statistical methods. Since its publication by Kocher et al. [1999] it has been proven to be a powerful tool for the extraction of secret information from cryptographic devices. While SPA is only based on visible trace variations caused by differences in the execution of the algorithm, DPA exploits trace information depending on data-dependent transistor switching. While the original presented techniques required high amounts of power or radiation records if simple countermeasures are present, newer methods reduced these significantly. Especially noticeable is correlation DPA as shown in Brier et al. [2004] and Mangard et al. [2007].

Countermeasures could be specialized logic styles and shielding to flatten emitted leakage profiles. Randomization and masking techniques help to counteract statistical methods and power consumption assumptions. Generally it is more difficult to efficiently fight statistical power analysis attacks. As shown in Schaumont and Tiri [2007] some countermeasure combinations may even be counterproductive.

4. POWER EMULATION (PE)

The wish for efficient power estimation methodologies led to the development of the power emulation technique, that is, the hardware-accelerated power evaluation of a given system-under-test. It became a promising alternative to simulation-based power profiling approaches and its hardware-based nature allows integration into embedded systems.

4.1. Principle of Power Emulation

Coburn et al. introduced the principle of *Power Emulation* (PE) in Coburn et al. [2005] as a state-of-the-art functional emulation using hardware-implemented power

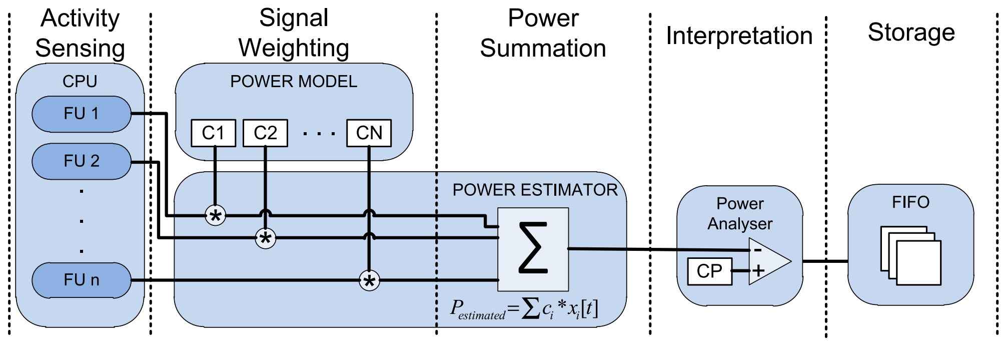


Fig. 2. Power emulation principle architecture (adapted from Krieg et al. [2011c]).

models. The functional emulation and power estimation process are executed concurrently during the runtime of the system.

As depicted in Figure 2 the generated approximated power values can be immediately used for further processing or saved as traces in case of power consumption evaluations.

4.2. Extended Power Emulation Unit

Our power evaluation platform is partially based on a high-level *power emulation unit* similar to the one introduced in Genser et al. [2009]. It comprises a direct implementation of the principle architecture depicted in Figure 2.

Its basic operation utilizes additive linear power macromodels as expressed in Eq. (1) to generate static and dynamic power consumption estimates. These values are generated for different system components by monitoring its power-relevant control signals x_i. For each member of the control signal set of size Nc a weight c_i and for each observed data-line a value d_i in a set of size Nd inside the macromodel is defined. The static power consumption is considered by coefficient c_0, resulting in a combined power model size of $Nc + Nd + 1$ coefficients that have to be implemented inside the emulation architecture.

$$\hat{P}[t] = \hat{P}_{sta} + \hat{P}_{dyn}[t] = c_0 + \sum_{i=1}^{Nc} c_i x_i[t] + \sum_{i=1}^{Nd} d_i x_i[t] \tag{1}$$

To enable the usage of this power estimation principle for DPA evaluation, its power model has to be extended for the usage of not only State-Dependent (SD) but also Data-Dependent (DD) signals. The original design of the PE unit was based on the analysis of the state of various system components by monitoring selected power-relevant control signals. This approach is only viable for pure power consumption estimation evaluations within a reasonable accuracy. A power model designed only under the consideration of a system's control flow is typically not usable for security evaluations because of the missing dependency between power profile and processed data. Therefore these power models have to be extended with basic signal switching information of data-processing architecture elements. Such an extended power emulation hardware module and characterization process has been presented in Krieg et al. [2011a].

In case of our chosen target architecture we selected both operand registers of its Arithmetic Logic Unit (ALU). The elements of these operands have to be preprocessed to extract only switching information. This approach allows to introduce data dependency into the power consumption model while resulting in a very similar power emulation result. This is achieved by data-line switching activity extraction to simulate the

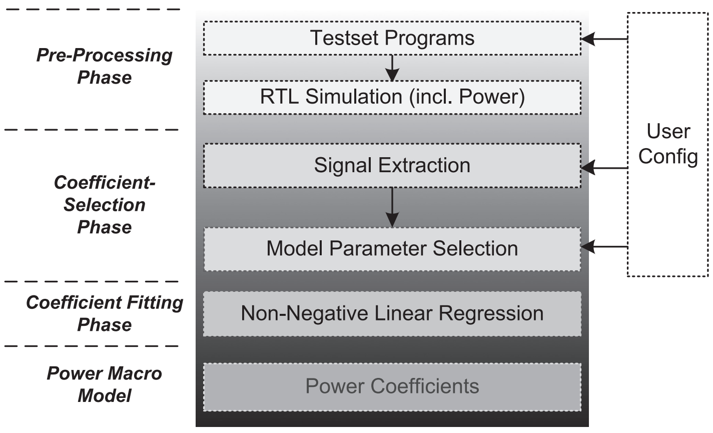

Pre-Processing Phase
Coefficient-Selection Phase
Coefficient Fitting Phase
Power Macro Model
Testset Programs
RTL Simulation (incl. Power)
Signal Extraction
Model Parameter Selection
Non-Negative Linear Regression
Power Coefficients
User Config

Fig. 3. Automated power characterization methodology for power evaluations.

current peak behavior of CMOS inverters. Such data-line weights can be either defined dynamically to investigate worst- and best-case scenarios, or by extraction of signal line capacitances (as provided by the RC extraction phase of common semiconductor back-end tools). Our generated power models have been evaluated and compared to gate-level power simulations and their accuracy has been confirmed to be similar to power models only concerning control signals. The evaluation of software side-channel leakage countermeasures can be enabled by scaling only these data-dependent model coefficients. In other words, leakage can be intentionally over- and underestimated to simulate well and less optimal routed designs resulting in different gate capacitances.

To enable our FPGA-based evaluation platform for simple runtime power tracing the power estimation unit has been additionally extended with an internal and configurable FIFO buffer. This memory allows the storage of the trace of a single encryption run without control interference of the cryptographic algorithm. After such an evaluation run has been finished its contents can be safely written to an external memory target.

4.3. Extended Power Characterization for DPA Evaluations

An automated power characterization methodology has been presented in Bachmann et al. [2010]. A similar approach has to be extended to enable the generated power models consisting of both state-dependent and data-dependent model coefficients. The power characterization flow as depicted in Figure 3 can be described by the following stages.

(1) *Preprocessing*. To gain a good coverage of all components of a given system-under-test, a set of microbenchmarking applications is simulated and power-profiled using state-of-the-art gate-level power estimation tools (Synopsys PrimeTimePX). This simulation process results in training sets containing signal activity and power estimation data.

(2) *Coefficient selection*. Our characterization approach applies several filters to the training set data retrieved during the previous stage to remove signals that show no (zero-activity filter), irrelevant, or redundant activity (cross-correlation filter). Basing on this filtered dataset power-relevant signals, that is, power model parameters x_i are determined according to their correlation to the gate-level power simulation data. The completely automated approach presented in Bachmann et al. [2010] only considers control-flow-related signals for power modeling. Therefore we extended this approach by the addition of data signals based on a user-defined

configuration. These data signals have to be preprocessed using an in-house VCD-file analyzer solution to extract only its switching activity.

(3) *Coefficient fitting*. The resulting parameters out of the selection process are now used for the determination of coefficients c_i by means of a model coefficients fitting process. This process is using a nonnegative linear regression technique. Finally a power model containing both state- and data-dependent coefficients is generated.

5. RELATED WORK

This work includes the application of a variety of technologies to gain a higher-level view of complex security problems. These techniques can be divided into three main groups. First, there emulation-based power estimation methodologies to retrieve accurate cycle-accurate power estimates at early stages of the design process. Second, security evaluations of cryptographic software and hardware implementations rely on the application of power analysis techniques as soon as possible during the design flow. Finally, fault injection platforms have been presented to confront hardware models or manufactured devices with eventual occurring faults.

5.1. Emulation-Based Power Estimation

In Coburn et al. [2005] a brief overview on the basic power emulation principle is given. Runtime improvements by power estimation hardware acceleration of about 10x to 500x compared to commercial power estimation tools are achieved. The authors also introduced strategies to minimize the hardware overhead caused by the emulation methodology. In later work this approach has been extended in Ghodrat et al. [2007] into a hybrid power estimation technique for complex SoCs. Power analysis times have been significantly reduced by this framework through the use of combined simulation and emulation techniques. A main advantage of hardware-assisted power estimation is the possibility of direct interpretation during runtime of a process. Such an approach, used for the power-aware process migration between cores has been presented in Bhattacharjee et al. [2008].

5.2. Early Power Analysis Techniques

The first approach comprises a purely software-based solution [Shumov and Montgomery 2010]. For extracting side-channel leakage information from a given program, it is executed and every instruction and processor state is traced. This information is fed into an easily extensible analysis module. Such analysis modules are directly implemented for the Microsoft Debugger API. The authors do not state any information about accuracy and simulation performance.

The second approach uses in-system knowledge provided by power consumption simulators using instruction set simulators. While promising high simulation speeds this approach is mostly limited to thoroughly profiled general-purpose architectures. Implementations of this approach were presented in den Hartog and de Vink [2005] and Thuillet et al. [2009]. Both solutions promise high simulation speeds, but do not present any hints about the accuracy of the used power profiles. Inaccurate power profiles could lead a software engineer into a sense of false security as the possibility of missed information leakage could be higher than anticipated. The implementation presented in Thuillet et al. [2009] uses several abstract power models to isolate possible sources of leakage. The presented approach has been designed for the Atmel AVR microcontroller series.

The third approach uses highly accurate gate- or even transistor-level simulations to identify critical parts of the circuit-under-test. This level of accuracy comes with high costs in terms of simulation speed and needed simulation equipment. The designer therefore has to choose between very short simulation times or reduction of the evaluation to the smallest possible leakage contributer. The former choice is mostly infeasible because it is of utter importance to know if an adversary could extract information using a high number of power traces. Works based on highly accurate analog simulations using SPICE simulators [Li et al. 2005; Regazzoni et al. 2007, 2009] and using Register-Transfer-Level (RTL) simulations [Bucci et al. 2007] for improving simulation speeds have been presented. However, the timely effort of offline power simulations remains a limiting factor for the trace count. The authors of Bucci et al. [2007] also note that RTL simulations may hide leakage contributors only visible after further design flow steps.

5.3. Fault Injection Platforms

To improve the reliability of high-safety applications several fault injection techniques have been introduced [Arlat et al. 1990; Jenn et al. 1994; Leveugle and Hadjiat 2003]. Faults can be injected into completely manufactured parts, for example, using radiation or during the design phase using hardware description manipulations. The latter can be divided into simulation- and emulation-based approaches depending if a model of the implementation is simulated or emulated on an FPGA platform.

Because high-level hardware descriptions can be simulated directly during the design phase, first fault injection tools for such hardware models were based on simulation techniques. One of these first simulation tools was MEFISTO specifically targeted on fault injection into VHDL models [Jenn et al. 2002]. A higher-level approach using the system-level description language SystemC has been shown in Rothbart et al. [2004]. A pure simulation-based technique has been complemented with automatized saboteur and mutant insertion strategies for the VFIT tool in Baraza et al. [2006]. Further improvements concerning the injection performance of simulation and emulation approaches have been presented in Valderas et al. [2007].

The need for higher fault injection coverage led to the heavy research activity in the field of emulation techniques. As shown in Leveugle [2002] emulation promises significantly higher injection rates and therefore enables the possibility to evaluate more possible fault configurations than using simulation. The usage of FPGAs for hardware prototyping also allowed to use novel reconfiguration techniques for fault injection as presented in Antoni et al. [2002]. Since the introduction of these techniques several improvements have been introduced in Zheng et al. [2008] and Daveau et al. [2009].

All these presented approaches have been developed for dependability evaluations, meaning that in most cases only single faults have been considered. As shown in Leveugle [2007] security evaluations need more complex fault models as intentional faults could happen at several positions at once.

5.4. Discussion

Existing work concerning the evaluation of SoC designs at early design stages had strong focus on a single problem. Therefore, physical- and emulation-based approaches have been introduced for dependability evaluations and simulation-based techniques have been the prime choice for security investigations. This results from the need for performance to satisfy statistical models covering random fault models and high behavioral accuracy for the investigation of circuit effects as gate switching or faults resulting from laser attacks.

As long as all parts of the system are investigated by the same entity, this is a valid procedure as the test engineers can use information about the software and hardware implementation. In large SoCs and in smart-cards this is usually not the case. The hardware is, for example, provided by a large semiconductor manufacturer and the

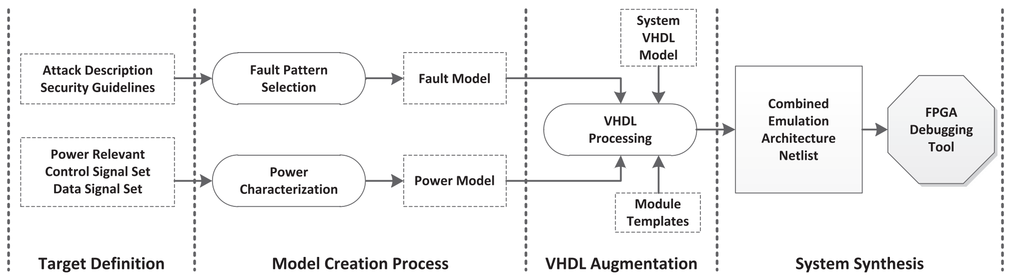


Fig. 4. Emulation hardware generation flow.

software is developed by a specialized company that targets a specific market sector. Especially in the smart-card section a wide range of rules has been defined for security evaluation to ensure the combination of the hardware and software fulfills a certain level of trust. Because the hardware manufacturer will try to hide the details of the secure implementation, it is not possible to provide such detailed information like a comprehensive simulation model to the customer. Our modular fault injection approach allows the hardware manufacturer to provide a debugging tool that abstracts the fault effect visible on software level from the hardware effects causing these effects.

Finally, the power consumption aspect has been completely disregarded in literature. In systems with a limited power budget a compromise has to be found between the most secure hardware architecture and a reasonable consumption behavior. A characterization and test environment to determine an optimized software and hardware implementation that fulfills both power and security targets is therefore strongly needed. To the best of our knowledge there is no comprehensive approach providing power analysis, fault analysis, and a complete power characterization flow, described in literature.

6. EVALUATION PLATFORM

The combination of fault injection and power emulation techniques opens a variety of new design analysis possibilities. Because of their hardware-accelerated nature they can be integrated in a combined form into an FPGA-based platform.

6.1. Emulation Hardware Generation Flow

As shown in Figure 4 the hardware-accelerated emulation architecture generation flow has been divided into the following phases.

— *Model Creation Phase.* In this phase a high-level fault event is mapped to a circuit-internal fault model. It is not necessarily run through during every fault emulation campaign. Additional power coefficient characterization has to be performed if significant hardware changes invalidated a previous power emulation implementation or if the flow is applied to a new design.
— *VHDL Augmentation Phase.* According to the chosen fault model and location, saboteurs have to be placed into the system to emulate a chosen fault effect. If necessary for further analysis steps, additional analysis modules also have to be placed during this stage.

These tasks have to be completed by the hardware designer to prepare the design and fault patterns for further distribution.

Model creation phase. The fault model itself can be divided into a low-level and a high-level view. The low-level fault model concerns the effects directly influencing the

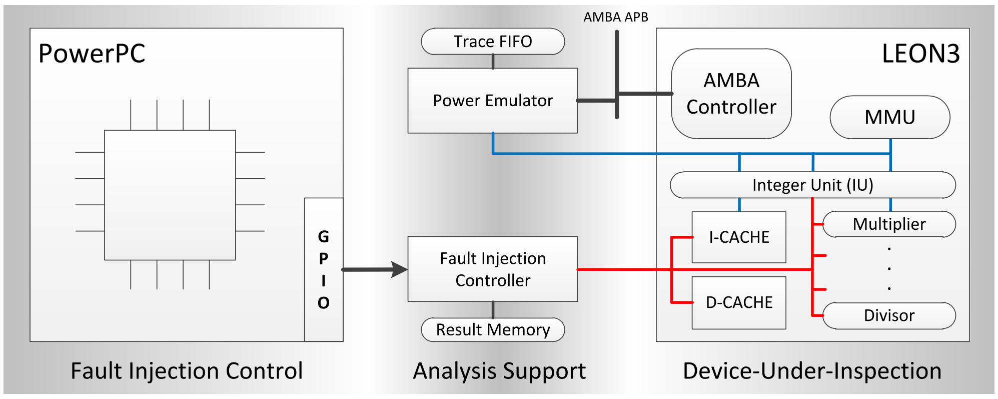


Fig. 5. Hardware-accelerated fault emulation architecture.

system on circuit level. Such fault models could be stuck-at, indetermination, delay, open, close, or bit-flip faults. A more detailed overview over such low-level fault models is given in Baraza et al. [2006]. High-level fault models can include many low-level effects resulting in a specific influence on normal program execution. These models are specifically important for security evaluations which include power or light attacks that lead to several faults happening at the same time [Leveugle 2007]. Currently we are not able to provide an automated approach to generate such high-level models and therefore good evaluations results rely on the experience of the test designer.

If the power consumption behavior of the targeted system changed significantly, the system has to be recharacterized to retrieve an updated power macromodel. The process has also been automatized using MATLAB for coefficient selection, characterization, and verification. The verification process compares the derived power model to cycle-accurate power simulation results for a chosen set of verification programs.

VHDL Augmentation Phase. After the selection of interesting fault models, fault locations can be determined. Onto each selected fault location a saboteur has to be placed to emulate the desired effect at the correct position. Selection, activation, and configuration signals of these saboteurs then have to be routed to the top level to be connected to the fault injection controller. Because of the complexity of modern processor designs this process has been completely automated using a VHDL parser application based on the VMagic VHDL code generator [Pohl et al. 2009]. The complete placement process is shown in Grinschgl et al. [2011a].

6.2. Fault Emulation Architecture

All parts of the proposed fault emulation platform are designed in a technology-independent way to allow for the application to FPGA evaluation systems of different manufacturers. In other words all augmentations are implemented on an RTL level to rely on existing implementation flows as provided by the evaluation board manufacturer. The basic hardware-accelerated fault emulation platform is depicted in Figure 5.

In case of a Virtex5-FXT-based FPGA system an integrated PowerPC processor can be used for fault injection control. The generalized GPIO interface allows to use any hard- or soft-core for control. Also an AMBA interface is available if an in-system solution is desired. The fault injection controller can be preprogrammed to listen for a specific Program Counter (PC) value. An internal counter is used to start the injection process when the PC of interest is called a predetermined amount of times.

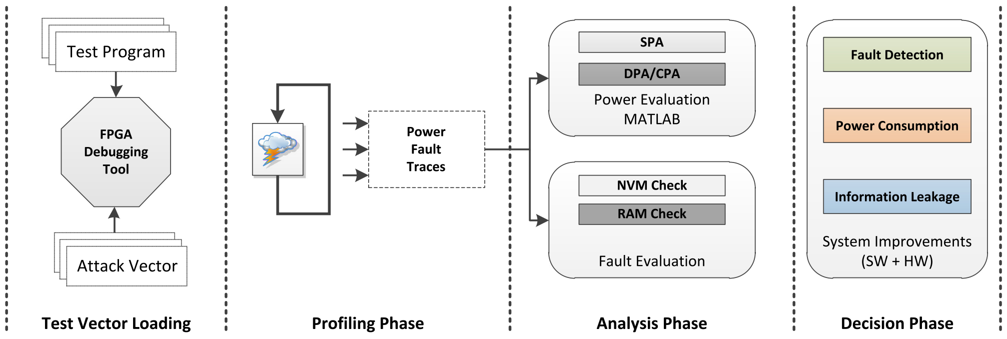


Fig. 6. Hardware-accelerated fault and power evaluation flow.

6.3. System Evaluation Flow

After all characterization and generation tasks have been executed and proper emulation architecture has been derived, the resulting netlist can be loaded onto a FPGA-based evaluation device. As shown in Figure 6 the following phases have to be passed through to gain information about system power consumption, leakage, and fault robustness.

— *Profiling phase*. In the profiling phase test programs are executed on the target platform while faults are continuously injected to gain knowledge about the possible execution effects.
— *Analysis phase*. All data gathered during the profiling phase is now thoroughly examined to detect reasons and develop countermeasures against undesired execution errors.
— *Decision phase*. Finally analysis results are used to select software or hardware measures to improve overall robustness of the target architecture.

Profiling Phase. The profiling phase comprises the main stage of the hardware-accelerated fault emulation process. A test set containing applications that would be considered a typical load during normal operation is executed on the evaluation platform. During this execution faults are injected using a fault injection controller and saboteurs routed during the placement phase. The fault injection controller can be preprogrammed to start injection at specific values of the processor program counter or directly controlled by an additional external processor.

Power profile tracing can be executed in two ways.

— *Runtime profile tracing using a PE unit with dedicated trace memory*. As described in Section 4.2 dedicated trace memory has to be provided to store intermediate power profile information. This memory has to be regularly read and saved to a persistent storage module for later analysis. This information can be easily converted to text files for usage with MATLAB scripts.
— *Power profile tracing using RTL simulation*. This would be the technique of choice if examinations have to done during a very early stage of the design process. The simulation model of the PE unit generates text files of a simple format that can be easily parsed and analyzed using MATLAB scripts. Encryption/decryption run trace separations have to be ensured using software commands, for example, by placement of NOP instructions.

Fault analysis tracing can also be done in two ways. First, the fault injection controller contains a checker interface to transfer data from dedicated checker modules to an external processor. These modules allow to evaluate which injected faults resulted into a direct operational fault. Second, additional analysis support modules can be routed into the design under inspection if a completely in-system solution is desired. Such support blocks could be checker or power data preprocessing modules to ease data interpretation or trigger modules to detect specific system operation states.

Analysis Phase. During the analysis phase recorded data is used to gain information about the power, dependability, and security characteristics of the targeted system. The derived power profile trace file of the executed test set programs is parsed using MATLAB scripts. This information then has to be divided into subtraces to reduce analysis effort.

In case of power analysis evaluations all en- or decryption traces have to be separated to enable DPA. The quality of the differential power analysis is highly dependent on the alignment of the analyzed traces. Therefore, eventual trace timing differences have to be corrected by cross-correlation techniques. To analyze the power analysis robustness of a given cryptographical software implementation the stored power profile traces can be processed using MATLAB scripts from the OpenSCA toolkit. This process has been completely automatized and results in attack attempts on all key bytes of the used key.

Fault emulation trace information recorded during execution can also be easily parsed using MATLAB to gain data matrices usable in efficiency and statistical analysis.

Decision Phase. After all analysis tasks have been performed the gained data can be used to derive changes of the targeted architecture. Such changes could include stronger error detection or recovery mechanisms or power optimizations to improve the available power budget. If several different fault-attack countermeasures are present it is also possible that their combination results in reduced robustness against power analysis attacks. In this case it is possible that other combinations or weaker mechanisms using different countermeasure techniques lead to a better countermeasure performance.

6.4. Automatized Fault Emulation

For real-world applications in the smart-card development and certification sector automatized fault injection campaigns are needed to ensure the coverage of a wide range of possible attack and device aging scenarios. To allow a similar test flexibility compared to slow fault simulations, modular control and fault injection elements as well as a high-level fault model representation are needed.

6.4.1. Modular Fault Injection Control Blocks. While simple small saboteur elements enable to generate faults at nearly every possible position inside the design, for exhaustive countermeasure testing it is also necessary to define points of time when an attack should be emulated. Such an attack time can be derived in the following ways.

— *Program Counter (PC).* The PC defines the current stage of the running program. By listening to this register an attack at a certain point of the program can be started.
— *Clock.* As communication programs can endure a significant amount of time, it is more reasonable to use a clock signal counter starting at a predefined point like a certain PC value.
— *Addresses.* If certain memory blocks have to be changed during an attack sequence it is necessary to examine system internal address and data buses. Access to the memory point of interest can be manipulated during transmission from or to the processor pipeline or cache.

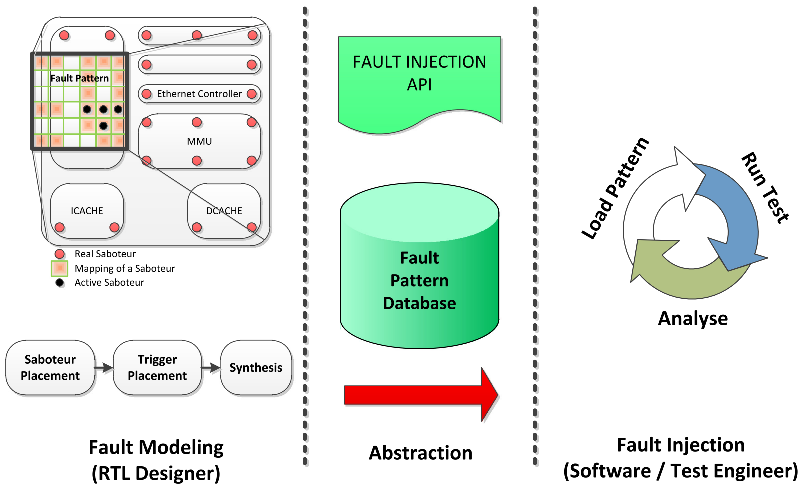


Fig. 7. High-level fault pattern approach.

To enable easy augmentation of a system-under-test so-called trigger modules are introduced that can be placed to a certain point of interest. As shown earlier most attack scenarios can be accomplished by listening to the PC register, clock signal, and address buses. The needed trigger functionality can be further reduced to two types of trigger modules: single-bit and multibit triggers. These trigger modules only output a binary signal to show if a given condition has been reached and therefore they can be combined by logic functions to create different fault activation conditions. All trigger modules are connected to the fault injection controller by shared buses to enable power-on configuration.

Single-bit trigger module. A single-bit trigger module contains a very simple counter structure and a configurable detection circuit. If the input signal changes a predefined amount of times, an output signal is raised to trigger further processing of other trigger modules or the fault injection controller.

Multibit trigger module. Similar to the single-bit trigger module this larger variant contains a counter structure and additional circuits to mask and compare the input value to a given target value. If necessary additional conditions can be compared, for example, the bus access status or specific access rights.

6.4.2. High-Level Fault Model Representation. The fault emulation system will also be provided to another company section or customer to test preproduction software on the given presilicon hardware design. As such a customer usually must not have internal information about the RTL description fault of model representations have to be provided. Such representations could be very specific fault patterns to emulate an attack shown in Section 7 or more general representations to simulation intentional memory changes.

Using control elements as presented in Section 6.4.1 the complete memory space of a given target application can be manipulated during runtime. This approach enables software developers with no access to the RTL description to do detailed fault-attack campaigns as depicted in Figure 7. These are necessary during certification phases or early evaluations of new secure operating systems.

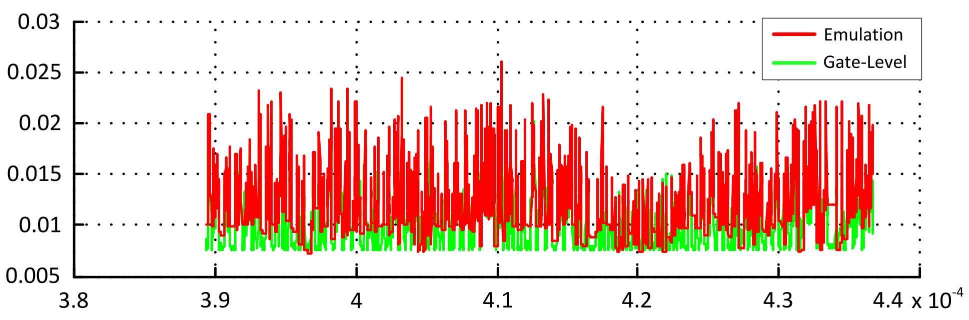


Fig. 8. Dhrystone benchmark: comparison gate-level to power emulation.

Table I. Fault Injection Performance Compared to Previous Approaches

Approach	Clock cycle	Inj. Faults	Time	Inj. Speed	Nor. Inj. Speed
	[ns]		[s]	1/[s]	1/[s]
[Civera et al. 2002]	50	100k	83	1205	603
[Lopez-Ongil et al. 2007] AE	16.8	129.75M	698	185888	276619
[Lopez-Ongil et al. 2007] PR	18.97	10k	1014	9.86	13
[Kafka 2008] DPR	10	-	-	12987	32468
[Kafka 2008] PRR	10	-	-	414.94	1037
This work	25	100M	46.76	2.17M	2.17M

7. EXPERIMENTAL RESULTS

To prove the usability of our approach we chose a widely used hardware platform and software implementation of the AES cryptography algorithm. For our hardware platform we selected an open-source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler [2010]. This processor has been synthesized using Xilinx ISE software and tested on the ML507 evaluation board also from Xilinx.

Our operating system tests have been executed using the SnapGear Linux distribution provided by Aeroflex Gaisler. Our analysis platform consists of MathWorks Matlab [2009b] on an six-core 3.2 GHz AMD Phenom-II machine using eight gigabytes of RAM.

The characterization process has been done using gate-level and power simulations using Synopsys PrimeTime provided by our industrial partner. This process resulted into a data-aware power macromodel containing 90 coefficients. For our selected 90nm production process library we achieved accuracies of lower than 4% average and lower than 15% RMSE cycle-accurate estimation errors. In Figure 8 a comparison of gate-level measurements with the results of power emulating the dhrystone benchmark is shown.

In Table I a performance overview is shown comparing the achievable injection speed of our approach to previously published systems. This evaluation concerns the pure injection of fault patterns into a given processor as shown in Grinschgl et al. [2011b].

The comparison is valid as these approaches only concern the injection of faults without any further processing of the resulting operating errors. Other solutions use autonomous emulation (AE), Partial Reconfiguration (PR), Partial Runtime Reconfiguration (PRR) and Direct Runtime Reconfiguration (DRR).

Table II. Fault Injection Evaluation of Parity-Based Error Detection Scheme - Data-Cache-Bus

		Detected Errors (DE)	DE [%]
Injected Faults	100000	-	-
2-bit Parity Scheme	-	74783	74.78
4-bit Parity Scheme	-	93705	93.7
5-bit Parity Scheme	-	96873	96.87

7.1. Dependability Evaluation

Fault injection is already widely used for the emulation of faults to test the efficiency of fault detection and recovery approaches. Existing solutions often do not consider the impact of such dependability improving hardware extensions on the system's power consumption profile. In power-constrained systems high power consumption peaks could lead to dangerous supply voltage drops that endanger operating reliability. To present the advantages of a multidisciplinary approach our LEON3 target system has been augmented with three different parity-based error detection schemes. These error detection mechanisms (2-, 4-, and 5-bit parity codes) have been added to the Memory Management Unit (MMU) to protect the instruction- and data-cache data-bus. These parity codes are checked inside the integer unit to ensure integrity of the read data. To implement our fault injection experiments we augmented this bus with a bus saboteur transparent under normal conditions. This saboteur allows the injection of multiple faults to every data-line routed through it. Fault patterns have been selected randomly and injected into a looping calculation. Data errors are detected if the parity code generated inside the MMU is different from the parity code generated inside the IU.

The results of such a fault injection campaign are shown in Table II and match our expectations concerning their operation principles. The 2-bit parity implementation performs significantly worse than stronger implementations. For the next evaluation step we could therefore select the 5-bit implementation for our power consumption explorations.

Another application for automated fault emulation campaigns would be timing investigations to test given designs for their robustness against device degradation effects. Signal delays caused by Negative-Bias Temperature Instability (NBTI) can be emulated by high-clocked delay elements inside the signal path [Krieg et al. 2011c].

Conclusion. Common error detection schemes using parity-based hamming codes have been thoroughly examined for their error detection efficiency. The proposed platform can also be used for the concurrent examination of fault resistance, power consumption, and delay behavior of a given RTL description.

7.2. Security Evaluation A (AES Software Implementation)

The nature of faults occurring during an attack scenario is very different to one of those resulting from natural causes. The reason is that these faults happen randomly while an adversary intentionally injects such faults in order to drive the attacked system into an unintended state. To prove the feasibility of our approach we chose a published attack on the AES symmetric cryptographic algorithm shown in Schmidt et al. [2009]. The authors presented a light attack on the AES-SBOX residing inside an programmable flash memory device. This attack is emulated by the placement of saboteurs on the memory data bus of our target architecture. These saboteurs are always activated when the memory region containing the AES-SBOX is read by our test program. This way all SBOX accesses return 0xFF as a result and are therefore

Table III. AES-SBOX Fault Emulation Results

1F	1B	17	13
1E	1A	16	12
1D	19	15	11
1C	18	14	10

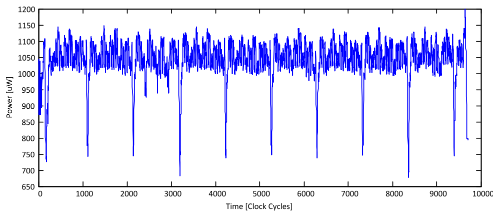
(a) Original Key

e0	1b	f7	08
e1	1a	f7	08
e2	19	f7	08
31	84	e9	1c

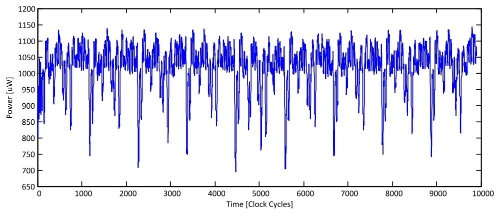
(b) Corrupted Output

1F	1B	17	13
1E	1A	16	12
1D	19	15	11
1C	18	14	10

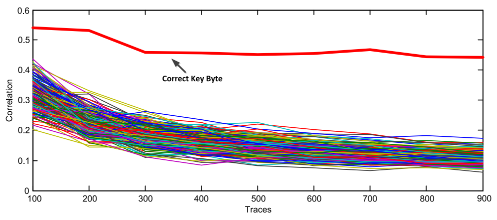
(c) Calculated Key



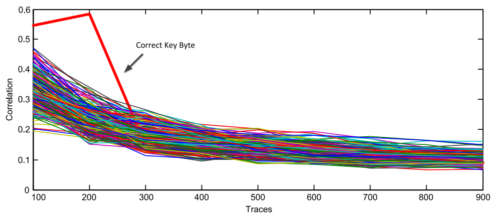
(a) single AES trace without any countermeasure application



(b) single AES trace using a light clock-gating strategy



(c) DPA attack on unmodified architecture



(d) DPA attack on architecture containing light countermeasure

Fig. 9. Tracing and differential power analysis (DPA).

equivalent to accesses to a device erased using ultra-violet light. The resulting wrong intermediate results can then be used to calculate the complete key used for this AES encryption. In Table III the input key of the AES encryption (a), the corrupted AES output (b), and the calculated key after the fault injection process (c) are shown.

By using a data-aware power estimation macromodel different power analysis types are also enabled. Figures 9(a) and 9(b) show cropped power traces containing one AES encryption run with and without influence of a simple clock-gating-based countermeasure as described in Krieg et al. [2011b]. To reduce power profile artifacts caused by task switching done by the operating system kernel, these figures have been recorded during RTL simulation. For emulation a 32000-sample FIFO has been implemented to allow the storage of a single trace in hardware for further analysis. The evaluation software has to read out this data after each tracing run to store large amounts of investigation data. A performance comparison of all applied techniques, as determined in Krieg et al. [2011a], is shown in Table IV. Figures 9(c) and 9(d) give an overview how successful a DPA attack on the unmodified and augmented architecture would be. Clock gating has been described in literature as a measure to increase profile noise and to decrease interprofile correlation [Benini et al. 2003]. In this case the chosen countermeasure would not have a significant impact on power consumption but its effectiveness against power analysis would have to be improved.

Conclusion. Therefore additional hardware used for improved circuit reliability can also be used to improve fault attack robustness. Simple manipulations of the clock signal provide only weak protection against power analysis attacks.

Table IV. Tracing Performance: RTL Simulation vs. FPGA Emulation
(taken from Krieg et al. [2011a])

	Traces	Time [sec]	Speedup
Direct RTL Simulation	100	3850.05	-
FPGA Emulation (w/o OS-overhead)	100	0.0258	149226
FPGA Emulation (w/ OS-overhead)	100	43.13	89.27

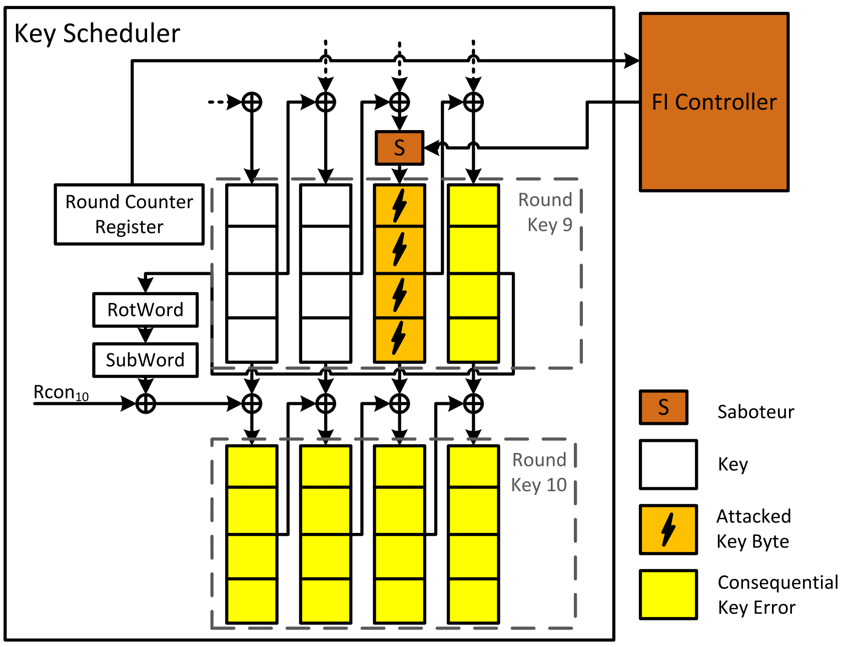


Fig. 10. Structure of the AES core fault-attack mechanism.

7.3. Security Evaluation B (AES Hardware Implementation)

In Takahashi et al. [2007] the authors describe a differential fault attack on the key schedule of an AES implementation. The authors claim that using this method it is possible to extract a complete 128-bit secret key using only seven pairs of correct and faulty cipher-texts without using brute-force search.

A reduced variant proposed in Takahashi et al. [2007] only needs two pairs of correct and faulty cipher-texts to extract 48 bit of the AES key. For this work we will show how to evaluate a given AES hardware block for its robustness against such a reduced fault attack.

Target of this approach is the key schedule mechanism, which can be manipulated at three different points as shown in Figure 10. The first point to be attacked is the 3rd 32-bit column of the 9th round key. Another possibility would be to attack 2nd 32-bit column and the 1st 32-bit column of the 9th round key. Both scenarios result in corrupted cipher-texts that can be used for the calculation of the final AES key. The following process is defined by seven rules as proposed in Takahashi et al. [2007] and does not require further attacks on the hardware.

For this case study we chose to attack the 3rd column of the AES calculation. For our target we chose an open implementation of the AES algorithm [OpenCores 2011]. As seen in Figure 10 we placed a saboteur into the 3rd column key generation mechanism. This saboteur is controlled by the Fault Injection Controller(FIC). The FIC has also been connected to the internal AES round counter register. This way the fault injection

Table V. Two Plain-Texts and Its Corresponding Cipher-Text Pairs

00	44	88	cc
11	55	99	dd
22	66	aa	ee
33	77	bb	ff

(a) Plain-text 1

69	6a	d8	70
c4	7b	cd	b4
e0	04	b7	c5
d8	30	80	5a

(b) Cipher-text 1

29	2a	71	fc
6b	c0	63	1b
27	d1	1a	c2
06	ec	fc	92

(c) Faulty cipher-text 1

00	04	08	0c
01	05	09	0d
02	06	0a	0e
03	07	0b	0f

(d) Plain-text 2

0a	41	f1	c6
94	6e	c3	53
0b	f0	94	ea
b5	45	58	5a

(e) Cipher-text 2

4a	01	67	92
3b	7c	d4	fc
e6	4f	39	ed
44	99	24	92

(f) Faulty cipher-text 2

Table VI. Fault Attack and Key Calculation Results

00	04	08	0c
01	05	09	0d
02	06	0a	0e
03	07	0b	0f

(a) Key

54	f0	b0[a]	1e[b]
99	85	91[a]	2e[b]
32	57	47[a]	3d[b]
d1	68	3c[a]	ee[b]

(b) Corrupted 9th Round Key

54	f0	10	be[c]
99	85	93	2c
32[c]	57[c]	ed	97[c]
d1[c]	68	9c	4e[c]

(c) Correct 9th Round Key

[a]Attacked Corrupted Key Bytes
[b]Consequentially Corrupted Key Bytes
[c]Calculated key bytes after fault attack

process can be automatically activated when this register reaches the 9th round. During the fault attack random bits of the targeted round key column are flipped.

Table V shows plain-texts and their corresponding cipher-text pairs with and without manipulation. After the fault attack using methods described in Takahashi et al. [2007] the following key bytes ($K_{0,3}$, $K_{2,0}$, $K_{2,1}$, $K_{2,3}$, $K_{3,0}$, $K_{3,3}$) have been calculated as shown in Table VI.

Conclusion. In this section the application of a direct manipulation of a cryptographic hardware module has been shown. The saboteur-based approach allows the testing of general crypto-implementations as long as the RTL-level hardware description is available to the test engineer. The general applicability of our automatized fault injection approach has been proven using an open available AES module and a nonimplementation specific hardware attack described in literature.

8. CONCLUSION

This article presents a novel FPGA-based platform and methodology for reliability and security evaluations of smart-card and general-purpose processor implementations. Through the combination of fault injection and power emulation techniques efficient hardware/software cosimulation of cryptographic software and hardware is enabled. The instant evaluation of power consumption and fault resistance behavior allows for the design of novel architectures under the consideration of power and security constraints. The effectiveness of our proposed hardware-accelerated fault emulation platform has been shown using software and hardware implementations of the symmetrical AES algorithm. Novel modular trigger and fault injection mechanisms allow for the efficient execution of long-running injection campaigns.

The efficient modeling and implementation of fault attacks will be play a vital role during the design of future secure architectures and certification for high security standards.

Our future work includes the usage of the proposed platform to examine new smartcard architectures for possible weaknesses. It also will be used to identify effective modifications for general-purpose architectures to improve their reliability and fault attack robustness. The combined approach will also allow for a detailed trade-off analysis of different error detection and recovery mechanisms considering both power and security constraints.

ACKNOWLEDGMENTS

We would like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

REFERENCES

AEROFLEX GAISLER. 2010. LEON3 processor. http://www.gaisler.com/.

ANTONI, L., LEVEUGLE, R., AND FEHER, M. 2002. Using run-time reconfiguration for fault injection in hardware prototypes. In *Proceedings of the 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'02)*. IEEE, 245–253.

ARLAT, J., AGUERA, M., AMAT, L., CROUZET, Y., FABRE, J.-C., LAPRIE, J.-C., MARTINS, E., AND POWELL, D. 1990. Fault injection for dependability validation: A methodology and some applications. *IEEE Trans. Softw. Engin. 16,* 2, 166–182.

ARLAT, J., AGUERA, M., AMAT, L., CROUZET, Y., FABRE, J., LAPRIE, J., MARTINS, E., AND POWELL, D. 2002. Fault injection for dependability validation: A methodology and some applications. *IEEE Trans. Softw. Engin. 16,* 2, 166–182.

BACHMANN, C., GENSER, A., STEGER, C., WEISS, R., AND HAID, J. 2010. Automated power characterization for run-time power emulation of SoC designs. In *Proceedings of the Euromicro Symposium on Digital System Design (DSD'10)*. 587–594.

BAR-EL, H., CHOUKRI, H., NACCACHE, D., TUNSTALL, M., AND WHELAN, C. 2006. The sorcerer's apprentice guide to fault attacks. *Proc. IEEE 94,* 2, 370–382.

BARAZA, J., GRACIA, J., GIL, D., AND GIL, P. 2006. Improvement of fault injection techniques based on VHDL code modification. In *Proceedings of the IEEE 10th International High-Level Design Validation and Test Workshop*. IEEE, 19–26.

BARAZA, J. C., GRACIA, J., GIL, D., AND GIL, P. J. 2002. A prototype of a vhdl-based fault injection tool: Description and application. *J. Syst. Archit. 47,* 10, 847–867.

BENINI, L., MACII, A., MACII, E., OMERBEGOVIC, E., PRO, F., AND PONCINO, M. 2003. Energy-Aware design techniques for differential power analysis protection. In *Proceedings of the 40th Annual Design Automation Conference (DAC'03)*. ACM Press, New York, 36–41.

BHATTACHARJEE, A., CONTRERAS, G., AND MARTONOSI, M. 2008. Full-System chip multiprocessor power evaluations using fpga-based emulation. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'08)*.

BRIER, E., CLAVIER, C., AND OLIVIER, F. 2004. Correlation power analysis with a leakage model. *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)*. Lecture Notes in Computer Science, vol. 3156, Springer, 135–152.

BUCCI, M., LUZZI, R., MENICHELLI, F., MENICOCCI, R., OLIVIERI, M., AND TRIFILETTI, A. 2007. Testing power-analysis attack susceptibility in register-transfer level designs. *Inf. Secur. 1,* 3, 128–133.

CIVERA, P., MACCHIARULO, L., REBAUDENGO, M., REORDA, M., AND VIOLANTE, M. 2002. Exploiting circuit emulation for fast hardness evaluation. *IEEE Trans. Nucl. Sci. 48,* 6, 2210–2216.

COBURN, J., RAVI, S., AND RAGHUNATHAN, A. 2005. Power emulation: A new paradigm for power estimation. In *Proceedings of the Design Automation Conference (DAC'05)*. 700–705.

DAVEAU, J., BLAMPEY, A., GASIOT, G., BULONE, J., AND ROCHE, P. 2009. An industrial fault injection platform for soft-error dependability analysis and hardening of complex system-on-a-chip. In *Proceedings of the 20th IEEE International Reliability Physics Symposium*. IEEE, 212–220.

DEN HARTOG, J. AND DE VINK, E. 2005. Virtual analysis and reduction of side-channel vulnerabilities of smartcards. In *Formal Aspects in Security and Trust*. Springer, 85–98.

GENSER, A., BACHMANN, C., HAID, J., STEGER, C., AND WEISS, R. 2009. An emulation-based real-time power profiling unit for embedded software. In *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS'09)*. 67–73.

GHODRAT, M., LAHIRI, K., AND RAGHUNATHAN, A. 2007. Accelerating system-on-chip power analysis using hybrid power estimation. In *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC'07)*. 883–886.

GRINSCHGL, J., KRIEG, A., STEGER, C., WEISS, R., BOCK, H., AND HAID, J. 2011a. Automatic saboteur placement for emulation-based multi-bit fault injection. In *Proceedings of the 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC'11)*. 1–8.

GRINSCHGL, J., KRIEG, A., STEGER, C., WEISS, R., BOCK, H., AND HAID, J. 2011b. Modular fault injector for multiple fault dependability and security evaluations. In *Proceedings of the 14th Euromicro Conference on Digital System Design (DSD'11)*. IEEE, 550–557.

GUNNEFLO, U., KARLSSON, J., AND TORIN, J. 2002. Evaluation of error detection schemes using fault injection by heavy-ion radiation. In *Proceedings of the 19th International Symposium on Fault-Tolerant Computing (FTCS'02)* Digest of Papers. IEEE, 340–347.

JENN, E., ARLAT, J., RIMEN, M., OHLSSON, J., AND KARLSSON, J. 1994. Fault injection into vhdl models: The mefisto tool. In *Proceedings of the 24th International Symposium on Fault-Tolerant Computing (FTCS'94)* Digest of Papers. IEEE, 66–75.

JENN, E., ARLAT, J., RIMÉN, M., OHLSSON, J., AND KARLSSON, J. 2002. Fault injection into VHDL models: The MEFISTO tool. In *Proceedings of the 24th International Symposium on Fault-Tolerant Computing (FTCS'02)* Digest of Papers. IEEE, 66–75.

KAFKA, L. 2008. Analysis of applicability of partial runtime reconfiguration in fault emulator in Xilinx fpgas. In *Proceedings of the 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS'08)*. IEEE, 1–4.

KOCHER, P., JAFFE, J., AND JUN, B. 1999. Differential power analysis. In *Proceedings of the Annual Cryptology Conference (CRYPTO'99)*. Springer.

KRIEG, A., BACHMANN, C., GRINSCHGL, J., STEGER, C., AND WEISS, R. 2011a. Accelerating early design phase differential power analysis using power emulation techniques. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST'11)*. 81–86.

KRIEG, A., GRINSCHGL, J., STEGER, C., WEISS, R., BOCK, H., AND HAID, J. 2011b. Run-time FPGA health monitoring using power emulation techniques. In *Proceedings of the 54th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'11)*. IEEE, 1–4.

KRIEG, A., GRINSCHGL, J., STEGER, C., WEISS, R., AND HAID, J. 2011c. A side channel attack countermeasure using system-on-chip power profile scrambling. In *Proceedings of the 17th International IEEE Online Testing Symposium (IOLTS'11)*. IEEE, 222–227.

LEVEUGLE, R. 2002. Fault injection in vhdl descriptions and emulation. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*. IEEE, 414–419.

LEVEUGLE, R. 2007. Early analysis of fault-based attack effects in secure circuits. *IEEE Trans. Comput. 56*, 10, 1431–1434.

LEVEUGLE, R. AND HADJIAT, K. 2003. Multi-level fault injections in vhdl descriptions: Alternative approaches and experiments. *J. Electron. Test. 19,* 5, 559–575.

LI, H., MARKETTOS, A., AND MOORE, S. 2005. Security evaluation against electromagnetic analysis at design time. In *Proceedings of the 10th International High-Level Design Validation and Test Workshop*. 211–218.

LOPEZ-ONGIL, C., GARCIA-VALDERAS, M., PORTELA-GARCIA, M., AND ENTRENA, L. 2007. Autonomous fault emulation: A new fpga-based acceleration system for hardness evaluation. *IEEE Trans. Nucl. Sci. 54,* 1, 252.

MANGARD, S., OSWALD, E., AND POPP, T. 2007. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer.

OPENCORES. 2011. Aes_crypto_core. http://opencores.net/project,aes_crypto_core.

POHL, C., PAIZ, C., AND PORRMANN, M. 2009. vMAGIC: Automatic code generation for vhdl. *J. Reconfig. Comput. 2009*.

RAVI, S., RAGHUNATHAN, A., KOCHER, P., AND HATTANGADY, S. 2004. Security in embedded systems: Design challenges. *ACM Trans. Embed. Comput. Syst. 3,* 3, 461–491.

REGAZZONI, F., BADEL, S., EISENBARTH, T., GROSSSCHAEDL, J., POSCHMANN, A., TOPRAK, Z., MACCHETTI, M., POZZI, L., PAAR, C., LEBLEBICI, Y., AND IENNE, P. 2007. A Simulation-based methodology for evaluating the dpa-resistance of cryptographic functional units with application to cmos and mcml technologies. In *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS'07)*. 209–214.

REGAZZONI, F., CEVRERO, A., STANDAERT, F.-X., BADEL, S., KLUTER, T., BRISK, P., LEBLEBICI, Y., AND IENNE, P. 2009. A design flow and evaluation framework for dpa-resistant instruction set extensions. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES'09)*. Springer, 205–219.

ROCHE, T., LOMNÉ, V., AND KHALFALLAH, K. 2011. Combined fault and side-channel attack on protected implementations of aes. *Smart Card Res. Adv. Appl.*, 65–83.

ROTHBART, K., NEFFE, U., STEGER, C., WEISS, R., RIEGER, E., AND MUEHLBERGER, A. 2004. High level fault injection for attack simulation in smart cards. In *Proceedings of the 13th Asian Test Symposium*.

SCHAUMONT, P. AND TIRI, K. 2007. Masking and dual-rail logic don't add up. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES'07)*. 95–106.

SCHMIDT, J., HUTTER, M., AND PLOS, T. 2009. Optical fault attacks on AES: A threat in violet. In *Proceedings of the IEEE Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 13–22.

SEGALL, Z., VRSALOVIC, D., SIEWIOREK, D., YASKIN, D., KOWNACKI, J., BARTON, J., DANCEY, R., ROBINSON, A., AND LIN, T. 2002. Fiat-fault injection based automated testing environment. In *Proceedings of the 18th International Symposium on Fault-Tolerant Computing (FTCS'02)* Digest of Papers. IEEE, 102–107.

SHUMOV, D. AND MONTGOMERY, P. L. 2010. Side channel leakage profiling in software. In *Proceedings of the International Workshop on Constructive Side-Channel Analysis ans Secure Design (COSADE'10)*.

TAKAHASHI, J., FUKUNAGA, T., AND YAMAKOSHI, K. 2007. DFA mechanism on the aes key schedule. In *Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'07)*. 62–74.

THUILLET, C., ANDOUARD, P., AND LY, O. 2009. A smart card power analysis simulator. In *Proceedings of the International Conference on Computational Science and Engineering (CSE'09)*. 847–852.

TSAI, T., HSUEH, M., ZHAO, H., KALBARCZYK, Z., AND IYER, R. 2002. Stress-Based and path-based fault injection. *IEEE Trans. Comput. 48,* 11, 1183–1201.

VALDERAS, M., GARCIA, M., CARDENAL, R., ONGIL, L., AND ENTRENA, L. 2007. Advanced simulation and emulation techniques for fault injection. In *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE'07)*. IEEE, 3339–3344.

ZHENG, H., FAN, L., AND YUE, S. 2008. FITVS: A fpga-based emulation tool for high-efficiency hardness evaluation. In *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications (ISPA'08)*. IEEE, 525–531.

Received October 2011; revised January 2012; accepted March 2012

System Side-Channel Leakage Emulation for HW/SW Security Coverification of MPSoCs

Armin Krieg, Johannes Grinschgl,
Christian Steger and Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology, Austria
{armin.krieg, johannes.grinschgl,
steger, rweiss}@tugraz.at

Holger Bock, Josef Haid
Design Center Graz
Infineon Technologies Austria AG
{holger.bock, josef.haid}@infineon.com

***Abstract*—During recent years a tremendous number of embedded systems has been introduced into every person's household. Such systems cannot only be found inside non-critical applications like entertainment devices but also in safety or security critical implementations like smart-cards. The increasing complexity leads to the introduction of several different co-design techniques to enable the parallel design of the system's hardware and software. Especially concerning security evaluation procedures this may raise a problem of trust between the manufacturer of the hardware and the software if both are different entities. To enable a bridge between these two worlds, simulation and emulation-based approaches have been shown in literature and industry to provide abstracted information about fault-attack effects to the software developer. However, no fast and cost-effective approach is available to provide a metric about how much of a given secret is leaking from the device to its environment. Therefore, this paper proposes such a metric and an emulation-based methodology to enable an early estimation of side-channel leakage to a possible adversary. The effectiveness of our approach is shown using a common available system-on-chip implementation using an open-source standard-cell library for characterization and a FPGA-based emulation platform for demonstration.**

I. Introduction

Differential power and fault analysis attacks (e.g. as proposed in [1] and [2]) create a tremendous problem for software and hardware manufacturers of secure applications like smart-cards. Power analysis attacks have been counteracted by masking techniques and complex new logic styles to reduce data and control dependency of the device's power consumption. To reduce the risk of information leakage caused by unintended system behavior resulting of a fault-attack, redundant processor architectures and periodic software checks have been introduced. All these countermeasures have a significant impact on needed hardware resources and operation performance and therefore are in direct contradiction to cost and performance constraints demanded by the customers.

The result of this situation are compromises that provide a high-level of security at acceptable system cost and speed. To prove the resistance of this security reduced solutions, a large amount of simulation and emulation testing is needed. In case of intellectual property inside a general purpose system-on-chip architecture it has to be tested if secrets are not lost outside of the pre-tested secure hardware block. Another important problem arises if certain security aspects are pushed from the hardware to the software level, in which case the software developer needs a possibility to estimate if the chosen countermeasures are sufficient.

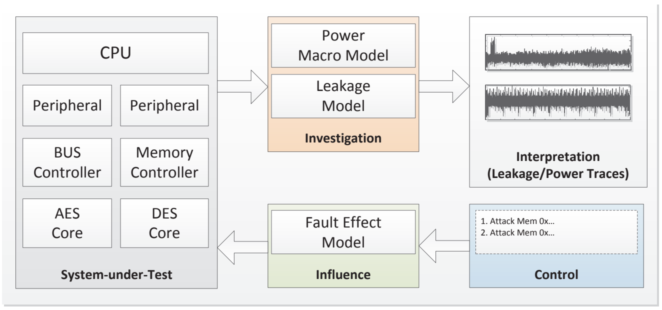


Fig. 1. System security emulation approach

Therefore, we propose an FPGA-based emulation platform using modular leakage sensor blocks to evaluate complex system-on-chips for eventual side-channels. This system approach allows to identify information leakage at any area of the complete data-path. This constitutes a strong advantage if not the complete device can be secured or if different entities are concerned with the implementation of the secure system. The main contributions of this work are:

- Extension of a power and fault-attack emulation approach with side-channel leakage estimation as shown in Fig. 1.
- Introduction of a novel methodology for the coverification of secure software and hardware in MPSoCs.
- A case study using a commonly available software and hardware implementation of the AES algorithm.

This paper is structured as follows. Section II provides a short overview about techniques for security hardware-software coverification. Followed by Section III introducing a novel side-channel leakage emulation flow. In Section IV the current state-of-the-art concerning simulation and emulation-based techniques for evaluation of secure systems is briefly reviewed. The efficiency of the proposed approach is experimentally investigated in Section V. Finally, our results are concluded and some details about our future work are given in Section VI.

II. Emulation Methodology for Secure Applications

Research on early side-channel evaluation techniques has been concentrated on hardware implementations of cryptographic algorithms like AES or DES. This resulted in highly optimized and secured crypto-modules while the surrounding architecture has remained fairly untouched. Large buses and memories are usually protected by simple masking techniques, encryption and physical shields to avoid probing [3]. To allow correct and secure operation also during fault-attacks, several standard tests have been introduced in the security evaluation. Both, the provider of the system software and of the hardware often strive to reach high certification levels, resulting in long and expensive evaluations of certified test labs. In case of found problems these tests have to be repeated resulting in longer time to market, increased costs and successively in a failed product.

Therefore, it is of vital importance to find deficiencies as soon as possible during the design phase and long before physical devices are available. At this development phase no physical side-channel measurements are possible, so developers have to rely on either simulation or emulation models. Additionally the hardware provider will not provide accurate information about system-internals for security and certification reasons. Basically there are two possible ways to solve this verification problem, first a detailed simulator or second a emulation system can be provided to the software manufacturer. Already emulation systems are provided to enable early software debugging, such systems can therefore be augmented with security relevant models to extent their functionality.

The advantage of emulation-based techniques in comparison to pure simulation models are manifold:

- High evaluation performance: Real-time execution with clock frequencies of the final product.
- Evaluation accuracy: The emulation netlist can be based on the same hardware description as the final device.
- Low costs: Existing debugging tools using FPGA-based functional emulation can be reused and additional functionality integrates into existing test-flows.

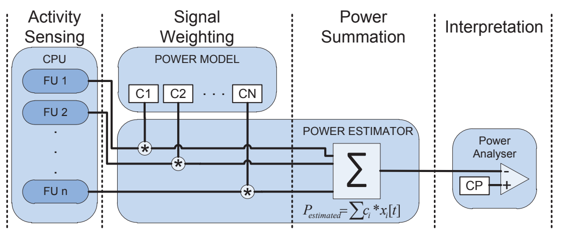


Fig. 2. Emulation-based power estimation flow (adapted from [4])

Because of the similarity of the basic system evaluation challenges the function principle of a power emulation approach is shown in Fig. 2. The power consumption behavior is determined through a characterization process and the context between control signal behavior and consumption stored inside a power macro model [4]. This way a cycle-accurate monitoring of the device's power consumption is enabled during runtime. Our proposed side-channel leakage flow, as described in the next section, will use such an abstracted model to provide an accumulated view on the system's information leakage while not providing detailed circuits information about internal structures.

III. Side-Channel Leakage Emulation Flow

Relying on the lessons learned from fault attack effect emulation for software verification purposes, a new side-channel leakage evaluation flow is proposed reducing a global problem to the examination of carefully chosen system elements. An adversary will most likely (very small semiconductor structures and strong shielding techniques result in very high costs for an adversary if one needs to access internal structures) only have access to the power consumption and emission of the device. In the optimum case these are not dependent on the control-flow and data used in the executed software. While they are caused by a large number of contributers, only these that constitute a possible security problem are of interest to the system and software engineer. Furthermore the path of a chosen secret and the behavior of the chosen logic elements concerning power consumption is well known.

A. Side-channel leakage modeling

Accurate modeling of side-channel information leakage is a complex field that has seen a large amount of research as shown in Section IV. Former work focused on the view of a cryptographic hardware developer that needs a very accurate model of the generated data-dependent power consumption. According to [5] the power consumption of CMOS circuits can be summarized by Equations 1 to 4.

$$P_{Charge} = p_t * C_L * V_{dd}^2 * f_{clk} \tag{1}$$
$$P_{ShortCicuit} = p_t * I_{SC} * V_{dd} * f_{clk} \tag{2}$$
$$P_{Leakage} = I_{Leakage} * V_{dd} \tag{3}$$
$$P_{total} = P_{Charge} + P_{ShortCircuit} + P_{Leakage} \tag{4}$$

While the first three terms contribute to the total power consumption of a CMOS circuit, only parts are relevant if only side-channel leakage is considered. The third term only describes the influence of the semiconductor leakage current $I_{Leakage}$ on the circuit consumption, it is therefore not data-dependent. If the first and second term are considered, supply voltage V_{DD} and operating frequency f_{clk} are constant and hence also not of concern. As the short circuit current I_{SC} is only technology dependent, only the coefficients p_t and C_L remain for consideration in a side-channel leakage model.

- p_t means the switching probability of the investigated logic circuit. This value is directly depending on the processed data, as it is dependent on the switching activity during clock signal changes.
- C_L means the load capacitance that has to be charged during switching of the logic gate. It is routing and implementation dependent and can be derived from the placed

and routed digital circuit during the RC extraction phase (if the standard cell library provides such information).

Hence, a side-channel leakage sensor has to at least consider switching activity of the observed data lines and the loading capacity of the observed logic gates. In Fig. 3 an intelligent sensor concept is presented that does not only detect data line bit toggling, but also provides a simple math function $F(x)$ for advanced analysis purposes. This approach for example allows to simulate masking mechanisms and to retrieve switching leakage that would result from the executed masked data. The result of this process are the number of toggled bits and a boolean value signaling if the unprocessed input was valid after comparison with the chosen filter data.

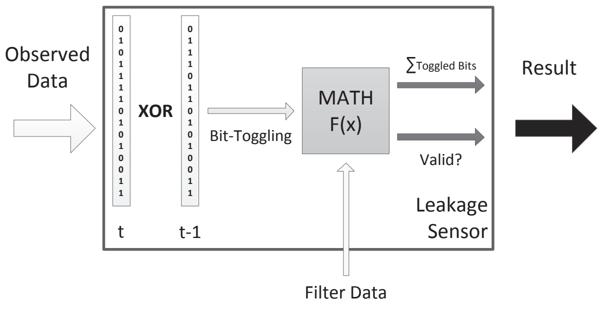


Fig. 3. Generic side-channel leakage sensor model

B. Leakage accumulation as a side-channel metric

Especially concerning cryptographic software verification an abstraction level between the hardware implementation and the emulation debugging platform is needed. To gain such an abstracted view on possible security leakage problems, the power analysis problematic is reduced to its core contributor, gate switching dependent power consumption. As described earlier leakage power is directly proportional to the capacitance of the placed gate. This behavior is not only valid for standard CMOS implementations, but every security logic style that relies on standard CMOS gates. For example, in the WDDL logic style, leakage highly depends on perfect routing to avoid glitches causes by asymmetries. As this style still uses CMOS gates, this problem could be modeled using the proposed intelligent sensor approach.

$$L = \sum_{i=1}^{Nb} c_i * x_i[t] \tag{5}$$

$$L_{accumulated} = \sum_{i=1}^{Nm} L_i \tag{6}$$

The retrieved side-channel leakage metric is described in Equations 5 and 6 where L denotes the amount of leakage for a specific sensor and $L_{accumulated}$ for the complete system. In case of a single sensor module, c_i means a coefficient of the leakage model and $x_i[t]$ the boolean value of an observed bit after processing at a given time.

C. Emulation methodology

The emulation methodology allows to evaluate cryptographic software and hardware implementation on different abstraction levels. Similar to fault emulation, modular sensor blocks are placed into the RTL code of the target device. Depending of the stage of the design process a generic or capacitance-based model is implemented inside the leakage accumulator as visualized in Fig. 4. These capacitance values are extracted from place&route data generated during the RC-extraction phase of the physical implementation process. After a model configuration has been chosen or extracted, the resulting RTL source can be directly simulated or fed into an FPGA synthesis process. This allows to do architecture and routing explorations before synthesizable and manufacturable source is available.

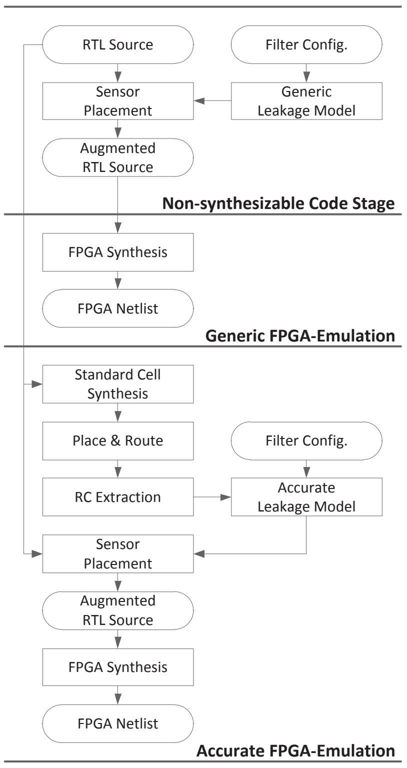


Fig. 4. Side-channel leakage emulation methodology

D. Emulation architecture

After a leakage sensor design has been chosen, these have to placed at critical data-path areas of the system. As shown in Fig. 5 this should include the complete path from memory to the processor internal pipeline registers. Additionally cryptographic hardware modules, like AES and DES en- and decrypters, can be included into the evaluation architecture to test existing masking techniques, or to simulate different masking strategies.

IV. Related Work

Side-channel attack evaluation techniques can be divided into two main groups depending on the used system models.

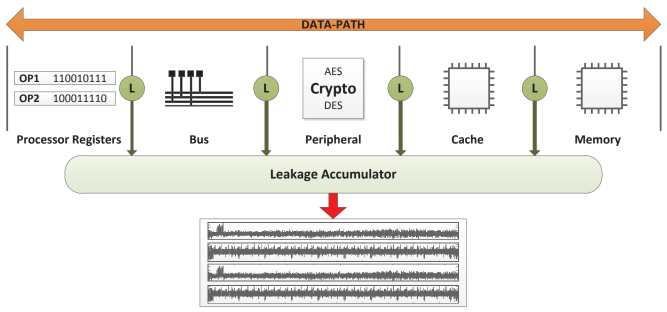


Fig. 5. Data-path side-channel evaluation architecture

The first group would be simulation techniques that have been researched for many years now resulting in a wide range of methodologies on different abstraction levels. The second group would be hardware-accelerated emulation methods to map system behavior to a FPGA-based evaluation platform.

A. Hardware Model Simulation

Tiri and Verbauwhede described simulation models for side-channel information leaks in a very detailed way in [6]. The authors described possible simulation inaccuracies caused by the wrong selection of circuit capacitances for generating a simulation model. These problems are of concern if an accurate model for DPA attacks is of interest, however if only the fact that information is leaked is investigated, the model is decoupled from the process of direct interpretation i.e. an DPA analysis. In [7] the authors describe RTL-level power-analysis attack testing without the inclusion of layout information. A highly theoretical platform-independent approach has been presented in [8], focusing on the security modeling challenges themselves. In [9] simulation performance is improved by the introduction of simplified capacitor charging models. All proposed techniques are relying on RTL or SPICE-level simulations and therefore cannot provide a suitable evaluation platform to an external entity.

B. Full System Simulation

In [10] a pure software-based analysis solution is presented. In this case the processor itself has been modeled for software verification purposes, however, the authors do not mention how a correct hardware model has been retrieved. The work presented in [11] relies on complete high-level modeling framework using the GEZEL hardware description language. The authors note that the usage of such complete new system-level design techniques needs novel tool chains, design cultures and practices. In [12] a full-system evaluation platform based on the SystemC modeling language has been introduced. While this has been a large step into the right direction it is still not based on the code base as the final product and therefore simulation inaccuracies will exists, especially if this language is not used for further evaluation purposes. In [13] a design flow and evaluation framework for novel systems with instruction set extensions has been introduced. Another pure software approach has been taken in the work shown in [14].

C. Emulation-based

State-of-the-art security system evaluation still heavily relies on simulation or investigation of manufactured devices. Only few literature is available describing emulation techniques for power analysis attack evaluation. The most similar approach compared to this work has been presented in [15], in which the authors presented high-level modeling technique based on toggle counting. Contrary to our proposal a hardware-verification method has been chosen, meaning that every logic cell has been extended with a 'toggle-counting-cell'. The resulting resource demands included a doubled need of lookup-tables (LUTs) and strong introduction of flip-flops. This increase makes this methodology not suitable for full-system emulation that includes a wide range of additional debug-functionality. A different approach has been proposed in [16] using an existing power-emulation characterization process to add data-dependent information into a pure control-flow power estimation flow. Hence, security relevant information can only be extracted using a statistical evaluation methodology, reducing emulation performance if applied to long-time tests.

D. Contribution

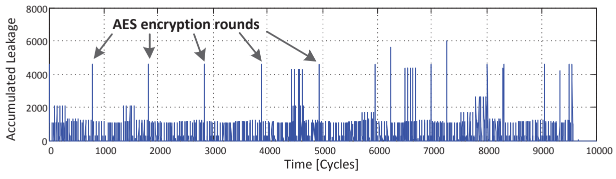
These evaluation techniques include the use of sophisticated differential power-analysis attack methodologies or rely on the application of large resource effort to gain information about the target application. Such toolkits are not usable in long-time testing environments because of needed computation performance and needed detailed hardware knowledge. This opens a wide variety of problems resulting from bad or even missing system security countermeasures. This is especially true considering software side-channel countermeasures that are not tested by experienced hardware engineers. Therefore, there exists a dangerous gap between hardware and software verification that could lead to unregarded side-channels resulting from bad system design. Our novel emulation approach utilizes intelligent modular sensor blocks to enable the observation of several different points inside the system, specifically the data-path. To the best of our knowledge there exist no emulation-based approaches for side-channel leakage estimation in literature.

V. Experimental Results

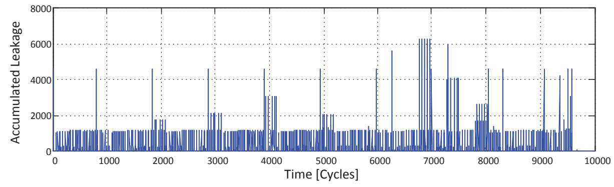
Our proposed methodology is implemented and demonstrated using an open source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler [17]. This system-on-chip example has been synthesized using Xilinx ISE software and is functionally tested on the ML507 evaluation board provided by Xilinx. All software evaluations are done using an open available and widely tested AES software implementation [18]. The hardware implementation has been taken from opencores.org and is therefore also generally available [19]. All physical capacitance data has been derived from a placed, routed and RC-extracted implementation using Cadence synthesis and P&R tools. The design has been generated using the open 45nm generic standard cell library provided by NanGate.

A. Software Verification (SWV)

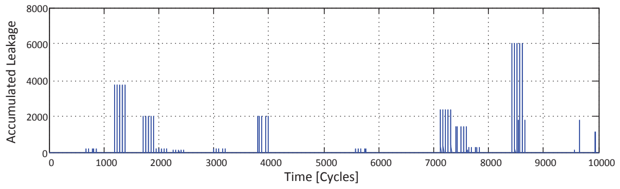
For software verification purposes several traces using different randomly chosen plaintexts are generated and the resulting leakage accumulation data can then be evaluated visually. Because the filtering process cannot directly distinguish between a transferred key and generic data, some key-independent data will also be visible inside the leakage trace. This behavior can be seen when comparing traces in Fig. 6a and Fig. 6b, as well as the regular peaks resulting from the unprotected processed key bytes. The key-independent trace fractions can also be visualized by filtering using any random key that is different from the actual used one as shown in Fig. 6c.



A.) AES encryption using random plaintext, filtered with correct key



B.) AES encryption using random plaintext, filtered with correct key



C.) AES encryption using random plaintext, filtered with random key

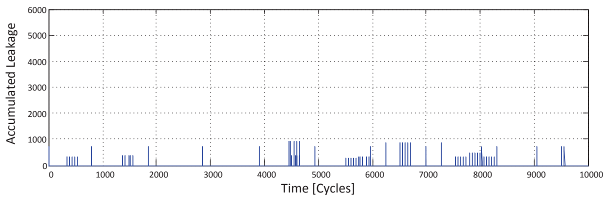
Fig. 6. AES software evaluation with correct and random keys

The modular sensor approach also allows for the location of strong leakage contributers. In Fig. 7a and Fig. 7b only the switching power for the two ALU input registers is shown. Because no bus is connected to these registers the resulting pin capacitances are relatively low and therefore their contribution is small as expected. For comparison the leakage contribution of the data-cache input bus is visualized in Fig. 7c.

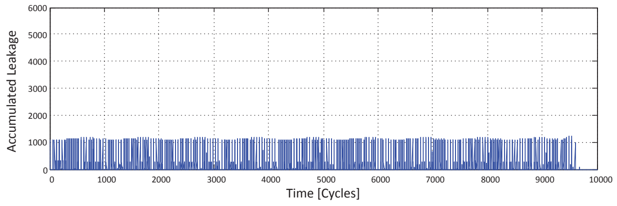
Although data-buses are the strongest contributers to switching power leakage, as described in literature, smaller fractions generated inside the processor pipeline cannot be ignored, especially if other system parts have been secured.

B. Hardware Verification (HWV)

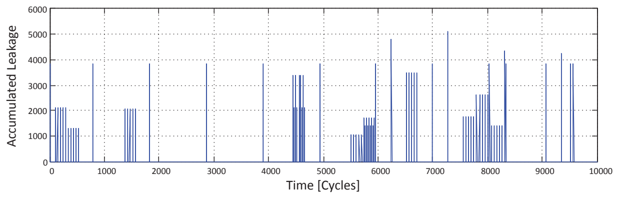
For the evaluation of an unsecured cryptographic module 16 byte-wide intelligent sensors have been placed onto an internal intermediate result register (SBOX-input). In Fig. 8a 100 encryption runs can be seen as the leakage result will peak when all sensors detect a correct key-byte. Because of the smaller sensor input width the amount of key-independent



A.) Leakage sub trace for sensor 3 – IU operand register 1



B.) Leakage sub trace for sensor 4 – IU operand register 2



C.) Leakage sub trace for sensor 5 – Data cache input

Fig. 7. AES software evaluation - submodule traces

data is higher than during the processor evaluations resulting in increased trace noise. The intelligent sensor approach allows to simulate a hardware masking technique, for example in Fig. 8b this internal result has been masked using a changing mask value. While the designer of the AES module now could feel safe about this chosen masking mechanisms, Fig. 8c shows that transfers of the key from the processor to the data-cache are strongly visible.

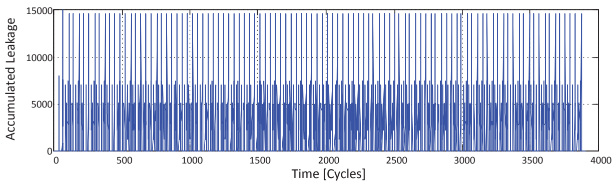
C. FPGA Implementation Results

As described earlier the proposed evaluation platform has been synthesized using Xilinx implementation tools to be used on the ML507 Virtex5 evaluation board. The results from this synthesis process are presented in Table I and show that the resource overhead is small enough to allow the placement of a large number of leakage sensors. The size of the data collector modules results of large adder structures that can be scaled down by the application of pipelining techniques.

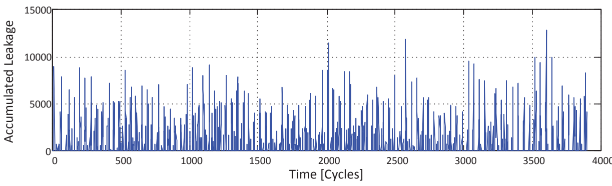
TABLE I
SYNTHESIS RESULTS - VIRTEX5 - BALANCED STRATEGY

Unit	Slice Registers	OH[a] [%]	LUTs	OH[a] [%]
Complete System	11568	-	38154	-
LEON3 Core	5611	48.5	12650	33.2
Data Collector SWV	23	0.19	3665	9.61
Data Collector HWV	27	0.23	1473	3.86
Leakage Sensor[b]	64	0.55	244	0.64
Leakage Sensor[c]	16	0.14	83	0.22

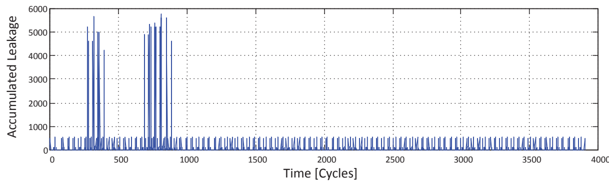
[a]Overhead compared to overall system resources
[b]32-Bit Sensor with XOR-function
[c]8-Bit Sensor with XOR-function



A.) AES hardware module – 100 traces without masking



B.) AES hardware module – 100 traces with masking



C.) AES hardware module – Processor data cache input leakage

Fig. 8. AES hardware module evaluation with and without applied masking

Even using unoptimized adder structures inside the data collector modules, the necessary hardware overhead for an FPGA implementation is significantly lower than comparable solutions described in literature, as shown in Table II. It has to be noted that the work described in [15] only a masked AES SBOX has been implemented with full logic coverage.

TABLE II
SYNTHESIS RESULTS - OVERALL OVERHEAD

Approach	Slice Reg. Overhead (OH)	OH[a] [%]	LUTs OH	LUTs OH[a] [%]
This work	1171	10.12	8216	21.53
Chen2009 [15]	744	-[b]	248	200

[a]Overhead compared to overall system resources
[b]Original design included no registers

VI. CONCLUSION

In this paper a novel side-channel leakage aware emulation methodology is introduced to enable the coverification of secure hardware and software for MPSoCs. High evaluation performance of FPGA-based investigation platforms is combined with leakage estimation based on post-place and route capacitance information taken from a standard cell digital system implementation flow. This technique allows to estimate possible information leaks inside the data-paths of hardened and unprotected multi-core processors during the design phase. Abstracted leakage information is provided to a possibly external software developer to also enable the evaluation of software countermeasures for unsecured hardware implementations. Combined with fault-attack effect emulation the functionality of countermeasures under attack conditions can be tested at early phases of the implementation process.

Our future work includes the research and implementation of efficient emulation-based techniques to enable fast and accurate coverification of secure MPSoCs. This includes the combination of power consumption estimation, fault-attack effect emulation and information leakage investigation into a single FPGA-based evaluation platform.

ACKNOWLEDGMENT

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749. We would also like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO*. Springer, 1999.
[2] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," *Advances in Cryptology, CRYPTO'97*, pp. 513–525, 1997.
[3] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer Verlag, 2007.
[4] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "A side channel attack countermeasure using system-on-chip power profile scrambling," in *IOLTS*. IEEE, 2011, pp. 222–227.
[5] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 4, pp. 473–484, 1992.
[6] K. Tiri and I. Verbauwhede, "Simulation models for side-channel information leaks," in *Proceedings of the 42nd annual Design Automation Conference*. ACM, 2005, pp. 228–233.
[7] M. Bucci, R. Luzzi, F. Menichelli, R. Menicocci, M. Olivieri, and A. Trifiletti, "Testing power-analysis attack susceptibility in register-transfer level designs," *IET 2007*, vol. 1, no. 3, pp. 128–133, 2007.
[8] F. Standaert, T. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," *Advances in Cryptology-Eurocrypt 2009*, pp. 443–461, 2009.
[9] D. Fujimoto, M. Nagata, T. Katashita, A. Sasaki, Y. Hori, and A. Satoh, "A fast power current analysis methodology using capacitor charging model for side channel attack evaluation," in *HOST*. IEEE, 2011, pp. 87–92.
[10] J. den Hartog and E. de Vink, "Virtual Analysis and Reduction of Side-Channel Vulnerabilities of Smartcards," in *Formal Aspects in Security and Trust*. Springer, 2005, pp. 85–98.
[11] V. Schaumont and I. Verbauwhede, "A component-based design environment for esl design," *Design & Test of Computers, IEEE*, vol. 23, no. 5, pp. 338–347, 2006.
[12] F. Menichelli, R. Menicocci, M. Olivieri, and A. Trifiletti, "High-Level Side-Channel Attack Modeling and Simulation for Security-Critical Systems on Chips," *Dependable and Secure Computing, IEEE Transactions on*, vol. 5, no. 3, pp. 164 –176, july-sept. 2008.
[13] F. Regazzoni, A. Cevrero, F.-X. Standaert, S. Badel, T. Kluter, P. Brisk, Y. Leblebici, and P. Ienne, "A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions," in *CHES 2009*. Springer, 2009, pp. 205–219.
[14] D. Shumov and P. L. Montgomery, "Side Channel Leakage Profiling in Software," in *COSADE 2010*, February 2010.
[15] Z. Chen and P. Schaumont, "Early feedback on side-channel risks with accelerated toggle-counting," in *HOST*. IEEE, 2009, pp. 90–95.
[16] A. Krieg, C. Bachmann, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "Accelerating early design phase differential power analysis using power emulation techniques," in *HOST*. IEEE, 2011, pp. 81–86.
[17] Aeroflex Gaisler, "LEON3 Processor," December 2010. [Online]. Available: http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53
[18] B. Gladman, "Implementations of AES (Rjindael) in C/C++ and assembler," December 2010. [Online]. Available: http://gladman.plushost.co.uk/oldsite/cryptography_technology/index.php
[19] H. Satyanarayana, "AES128," January 2012. [Online]. Available: http://opencores.org/project,aes_crypto_core

Characterization and Handling of Low-Cost Micro-Architectural Signatures in MPSoCs

Armin Krieg, Johannes Grinschgl,
Christian Steger and Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology, Austria
{armin.krieg, johannes.grinschgl,
steger, rweiss}@tugraz.at

Andreas Genser, Holger Bock,
Josef Haid
Design Center Graz
Infineon Technologies Austria AG
{andreas.genser, holger.bock,
josef.haid}@infineon.com

***Abstract*—In recent years the wide spread introduction of small embedded systems into every corner of everyday life lead to the strong need for highly reliable and secure computing machines. These machines now affect the safety of humans as well as the security of personal data and consequently money transactions. To ensure the integrity of these systems' operating state, several fault detection mechanisms have been developed to safely correct or stop unforeseen execution behavior. Because of the rise of battery or even field-supplied systems these mechanisms often heavily decrease available power budgets or lead to significantly increased production costs.**

Therefore, this paper introduces novel micro-architectural execution signature characterization and handling techniques for system-on-chip designs providing power estimation hardware. Existing power sensor infrastructure is reused to enable efficient system-state monitoring using micro-architectural hashes to cover a wide range of implemented system functionality. Reduced hashing implementations are characterized for their fault detection efficiency. This hardware-based approach provides a completely transparent solution to counteract faults resulting from emerging wear-out defects or intentional attacks on the execution integrity.

I. Introduction

Research on sophisticated fault detection mechanisms using on-line control flow monitoring has been ongoing for decades. The main driver behind this research has been computing machines for high-availability and security applications. Such threats to system integrity come from two major contributors. First, continuing semiconductor process integration leads to finer transistor structures and therefore to a higher probability of transistor defects. Second, criminal subjects try to gain secret information from secure devices by disturbing normal operation using sophisticated fault attacks.

In case of systems for highly dependable applications it is important to detect control flow changes because of faults resulting of environmental influences or manufacturing defects. These faults are of a random nature and their detection will usually lead to the activation of countermeasures to prevent erroneous actions that could result in loss of money or life. In case of high-security applications like smart-cards, fault detection is essential to identify attacks by an external adversary. The primary function is to prevent the loss of information and to hide the fact that the attack was successful. To guarantee a high fault detection coverage usually modular redundancy techniques are used. By duplicating or tripling processing units, even small control flow changes or data manipulations can be detected or corrected in case of a triple configuration. The disadvantage of such a fault detection approach are high implementation and manufacturing costs. In the high volume smart-card market such additional production costs are prohibitive.

Therefore, a strong need for control flow manipulation detection mechanisms providing a similar detection coverage as modular duplication while relying on a smaller resource footprint rose in the recent past. To gain an area effective implementation of a control flow observation mechanism, existing infrastructure has to be reused and simplified hashing implementations have to be found. Accurate on-line power estimation is also dependent on a view of the current control state of the system and therefore a hybrid monitoring structure is proposed. Hashing mechanisms are simplified by round and structure reduction and have to be characterized for the impact of these simplifications on fault detection efficiency.

The main contributions of this work are:

- Introduction of a novel control-signal signature hardware characterization strategy and hybrid power and fault monitoring infrastructure
- Integration of the proposed methodology into a state-of-the-art power profiling and functional emulation flow
- Case study using an augmented version of an open available system-on-chip implementation

This paper is structured as follows. Section II is giving a short introduction into control-state based hardware power estimation. In Section III common mechanisms for micro-architectural signatures are roughly described and the state-of-the-art concerning these signatures for general purpose architectures is briefly reviewed. Followed by Section IV introducing a novel transparent micro-architectural signature generation methodology. The efficiency of the proposed approach is experimentally investigated in Section V using a common general-purpose-processor system. Finally, our results are concluded and some details about our future work are given in Section VI.

II. Control-State based Hardware Power Estimation

Hardware power estimation has emerged as a rich technique in order to obtain power consumption information in a cycle-accurate manner. Having on-chip power information available to drive power management algorithms is becoming increasingly important for power-constrained embedded systems [1]. In particular in field-supplied systems it is mandatory to have power monitoring mechanisms available in order to track the power consumption of the system and to detect and to prevent the occurrence of harmful events (i.e., power peaks, supply voltage drops).

Additional hardware blocks enable the observation of internal system states, which are then fed to a power model implemented in hardware. Power estimates delivered by the power model can then be exploited by power management mechanisms. The principle of control-state inspection for system monitoring is similar to a signature-based fault detection approach as proposed in this work. Therefore a fusion of both evaluation techniques (power and fault) is possible to optimize resource efficiency. This step includes the relying on a joined activity sensor infrastructure and characterization process.

A. Power Model

The set of internal system states $\mathbf{x}$, each of which denoted by x_i and model coefficients c_i constitute the power model as depicted in (1)

$$P(\mathbf{x}) = \sum_{i=0}^{n-1} c_i x_i + e. \quad (1)$$

The power model's result gives power estimates $\hat{P}(\mathbf{x})$ corresponding to the linear combination of internal system states x_i and the model coefficients c_i. The difference of the power estimate $\hat{P}(\mathbf{x})$ to the real power consumption $P(\mathbf{x})$ is denoted by error e. The selection of a representative set of internal system states as well as the determination of the power model coefficients is performed during a power characterization process [2]. This process includes the execution of benchmarking programs using a gate-level description of the design to create an accurate power consumption abstraction.

B. Power Estimation Architecture

The power estimation architecture as depicted in Figure 1 consists of a number of power sensors that observe and map internal system states x_i to power model coefficients c_i. Power estimates delivered by the power sensors are summed up in the power accumulation unit, which finally provides the power estimate $\hat{P}(\mathbf{x})$.

The accuracy of the power estimate $\hat{P}(\mathbf{x})$ gathered from the power model depends on the number and quality of the observed system states. The approach is flexible in a way that the system state granularity at which the system is observed can be tailored to accuracy requirements of the power estimates. This is also advantageous for the coverage of control-flow based fault detection mechanisms by increasing the number of instruction-decode stage signals that are considered.

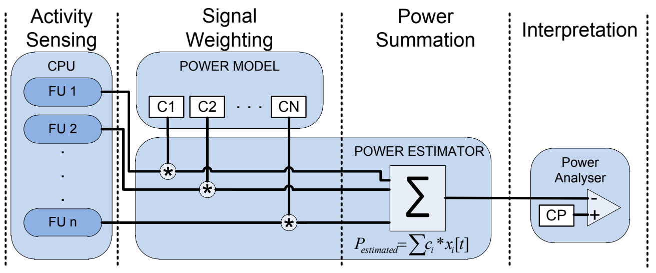


Fig. 1. Control signal based power estimation principle (adapted from [3])

III. Related Work

State-of-the-art research in the field of fault detection mechanisms using on-line control flow monitoring can be split into three main groups.

A. Software-based signature mechanisms

Purely software-based approaches mostly rely on the automatic or semi-automatic generation of code signatures to detect program flow changes caused by tampering or environmental influences. These signatures are directly embedded into the executed binaries. For the signature checking procedure itself processor internal resources have to be reused resulting in a high impact on execution performance.

Pure software solutions have been proposed using extensive redundancy as shown in [4] or by applying additional state checking to catch unforeseen execution control behavior as presented in [5] and [6]. An approach solely based on software signatures generated during compile time is proposed in [7]. The main disadvantage of these software-only approaches is their significant impact on execution performance. Furthermore, software checks could be manipulated by an adversary if such techniques are used in security critical applications.

B. Hardware-based signature mechanisms

Most commonly used hardware blocks for integrity checking are hardware monitors. Depending on the monitored unit these monitors can be implemented very efficiently. This advantage is counteracted if large memories are integrated to store precomputed signatures. Another problem concerning the state-of-the-art in this field is the selection of the monitored system region. Pipeline-only approaches may be insufficient for system-on-chips and the use of multiple monitors increases system complexity and decreases area efficiency.

To reduce the impact of the monitoring process on the operating performance and to gain direct access to hardware resources, control flow-monitoring approaches using dedicated monitoring hardware have been introduced. Such dedicated monitoring hardware could be watchdog-type modules as shown in [8] and [9] or smaller specialized monitoring circuits covering only selected parts of the system as presented in [10], [11] and [12]. Other approaches rely on modification of the processor pipeline to enable the observation of the control flow [13]. While being very effective in processor-only applications, it is often not sufficient for system-on-chips including a wide range of different processing units.

C. Hardware-Software co-design solutions

To ensure a wide detection coverage several co-design based approaches have been published. In this case the possibility to adapt both software support and hardware structure is used to reduce memory overhead and performance penalties. Such systems are generally based on application-specific instruction processors (ASIP) as published in several recent papers [14]–[16]. By design it is only possible to apply such techniques if the target architecture is either highly adaptable or the design process is at a very early stage enabling instruction set changes.

Another possibility would be to augment existing hardware to generate representative values to track control changes inside the system-under-test. Such reusable hardware blocks could be scan-out-chains as presented in [17]. This approach while having only a small impact on the complexity of the targeted system relies on periodic tests and therefore is not a real on-line testing solution. Another possibility would be fingerprinting techniques generating hash-values from the complete architectural state as published in [18]. The proposed fingerprinting approach while providing very accurate fault detection mechanisms leads to high demands on circuit bandwidth.

D. Contributions

Our proposed approach can be categorized into group number three supporting system-internal fault detection mechanisms as well as software controlled solutions. Compared to existing work our implementation does not rely on large architecture augmentations as necessary in group two and does not imply performance degradations of group one. It can be applied to any hardware containing state-dependent power estimation (PE) units without relying on large hash generation circuitry. This co-existence with PE hardware should also result in a very limited amount of additional needed area resources. Furthermore, our approach not only covers a microprocessor's pipeline but large parts of the entire system-on-chip to guarantee a wide fault detection coverage. The general applicability only relies on the availability of the RTL system description.

IV. Signature Mechanism for Resource Constraint Systems

In resource constraint systems like smart-cards huge memory, runtime or hardware overhead is prohibitive. Therefore, large control-flow graphs, embedded code signatures or extensive monitoring hardware cannot be employed to ensure correct system behavior. On the other hand the continuing integration of additional components into system-on-chip devices also adds a wide range of additional targets for attacks and points where degradation could lead to system failure.

Therefore, we propose a complete signature element selection and hardware description augmentation methodology using passive signature generation and comparison hardware. The goal of this approach is to provide high control flow security compared with low performance and area overhead. An overview of this proposed characterization and generation methodology is depicted in Figure 2.

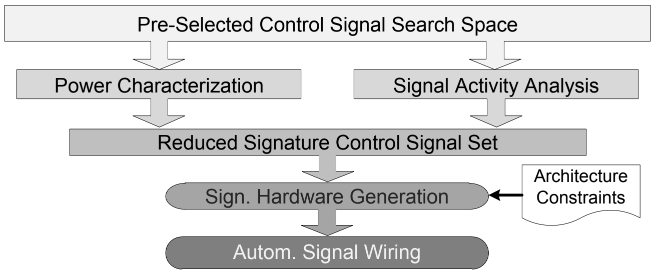

Pre-Selected Control Signal Search Space
Power Characterization
Signal Activity Analysis
Reduced Signature Control Signal Set
Architecture Constraints
Sign. Hardware Generation
Autom. Signal Wiring

Fig. 2. Signature architecture augmentation methodology

A. Control signal selection

This part of the signature implementation process is crucial to keep hardware effort low while enabling reliable and efficient system monitoring. The following two main criteria are important when selecting control signals for monitoring purposes:

- **Relevance**: Significant control signals should show strong activity in a large selection of standard applications. This requirement is also of importance considering control signal based power estimation to gain good estimation accuracy. Therefore, a power model characterization process will be employed to determine good candidates for hardware block signatures. A good example for such a characterization process is given in [2].
- **Determinism**: Signal activity must be directly dependent on the executed instruction sequence. In this case similar criteria are important for fault injection campaigns to simulate device degradation effects caused by high transistor activity in systems manufactured using very fine semiconductor technologies. Therefore RTL-simulation based activity analysis techniques will be used to determine good execution-signal correlation. The principle analysis techniques are similar to those shown in [19].

B. Signature type selection

Selection of a proper signature type is crucial to keep hardware overhead low without increasing the risk of collisions over an unacceptable level. Most hash signature algorithms, while being very robust against collisions and providing good coefficient mixing, need a significant amount of area in their hardware implementations. In case of system-internal execution signatures collisions are of lower concern than constraints on coefficient mixing. Therefore, the design can rely on hash algorithms with good avalanche behavior, meaning large output changes at small input changes, while being less optimal in means of collision resistance. Another important point when selecting a good hardware signature algorithm is calculation latency. Arora et. al. solved the problem of limiting maximum calculation latencies by adding hardware and limiting the maximum block size to 512 [20]. Still their MD4 and MD5 implementations had a significant impact on operating performance because the pipeline stalled until the hash generation is finished. An overview over small and fast hash functions for hash table lookup including MD4 is shown in [21]. To gain a lightweight memory-free hardware implementation only functions are considered that do not rely on coefficient tables. This excludes the simple SBOX-function that relies on large data tables while being computationally simple. All chosen signature implementations have been configured to result into a compression factor of 2:1. Meaning that the resulting hash value is only half as wide as the input vector. All hash implementations have been round reduced and simplified to retrieve a highly efficient hardware integration. Specifically a single-round implementation has been targeted to gain fast detection of execution variations.

For our exemplary implementations the following algorithms have been chosen:

CRC hashing: Well documented and automatically generatable for a wide variety of bit-widths are cyclic-redundancy-check (CRC) code generator-modules. For the VHDL code generation all even coefficients have been chosen to retrieve a constant hardware structure [22].

One-at-a-time hash function: The one-at-a-time hash algorithm shown by Jenkins et al. in [21] has been chosen because of its simple structure and because it can be simply implemented in hardware without additional execution latency. Compared to CRC one-at-a-time provides very good avalanche behavior while more hardware effort is needed.

MD5 secure hash: The well known MD5 hash algorithm has been reduced to retrieve a small and fast implementation. This reduction is achieved by only using one simplified round of MD5.

SHA secure hash: The basic structure of this signature implementation is very similar to the MD5-based one. Therefore, the same restrictions and characteristics apply for this low-cost hash function.

C. Signature segmentation

The possibility of fault location extraction from an execution signature is of vital importance for further fault analysis processes. Therefore, it has to be segmented depending on the available systems elements. After control signals have been grouped depending on their original location, individual signatures are generated and concatenated as shown in Figure 3. Such groups would be individual processor-cores in multi-core systems or system-on-chip elements like network controllers or caching control blocks. The selection of group members is done using a component-aware power characterization process to identify deterministic and relevant module control signals.

The efficiency of the segmented signature approach depends on the scalability of the hashing implementation and the granularity of the signature segments.

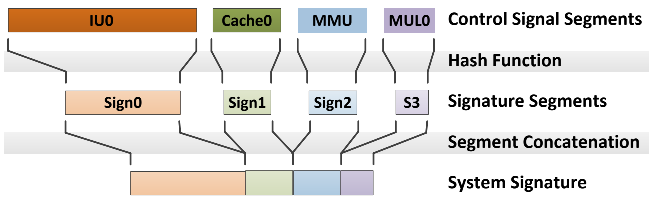


Fig. 3. Signature segmentation approach

D. Signature-based execution architecture

Depending on the available memory resources on-line or off-line signature checking techniques can be used.

Off-line techniques would include a software characterization process to generate execution signatures of targeted code regions. In this case a set of the calculated hashes has to be saved to the target device to be compared in hardware during regular operation. The advantage of this approach is the lack of multiple signature generator circuits as the reference is produced using a system emulator provided to the software developer. On the other hand large signature memories are needed to store the control flow history of the complete code execution.

On-line approaches include both, the reference and control signature generation, inside the target hardware. This can be implemented in a completely memory-free manner using a reference signature generation step during the fetch stage of the pipeline. Another possibility would be to use a system emulator for instruction characterization and in-system comparison at the end of the processor's pipeline. The memory-free approach will rely on a complex signature generation circuit predicting the future of the control signal state at a later stage. Instruction-only characterization constitutes a good compromise between needed circuit and memory requirements.

For our experimental evaluations an off-line approach using complete pre-calculated signatures has been chosen to demonstrate the feasibility of our technique. These signatures have been derived using a golden model run of the selected benchmark programs. The selected architecture is depicted in Figure 4.

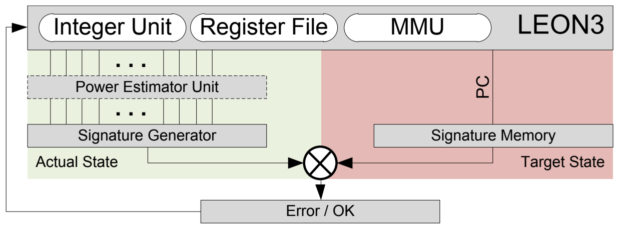


Fig. 4. Signature generation and comparison architecture

V. Experimental Results

The applicability of our approach is shown by using a widely known hardware platform based on an open source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler (www.gaisler.com).This system-on-chip example has been synthesized using Xilinx ISE software and tested on the ML507 evaluation board provided by Xilinx. This processor type constitutes not a classical example of a smart-card processor because of its complexity but it is still used to prove the general applicability of the chosen approach. Our analysis platform consists of MathWorks Matlab 2010b on a six-core 3.2 GHz AMD Phenom-II machine.

Using the semi-custom design flow and power simulation tools provided by our industrial partner a power macro model containing 63 coefficients has been derived. The power estimation module itself implements this power model using a three stage pipelined adder structure.

The following experimental evaluations of the signature generator modules have been done using 100000 randomly generated input vectors. To enable a common simulation environment all blocks have been implemented into a single testbench scaled for different input widths.

A. Signature performance

First the signature hardware selection has been evaluated to determine if every input change also results in a signature change. Especially when using small signature widths or reduced implementations suboptimal hash generator hardware will lead to increased amount of such collisions. Figure 5 shows the behavior of every signature implementation for different hash widths. It can be clearly seen that while one-at-the-time is the worst and CRC-based the best, SHA-based and MD5-based modules result in similar results.

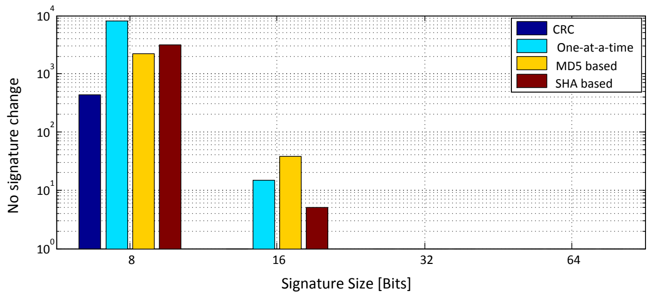


Fig. 5. No signature switching on input change

Second the amount of switched input bits compared to switching output bits has been evaluated. This ratio is supposed to be as low as possible, meaning that an input signal change results into a large change of the output value. The results are shown in Figure 6 concluding to a similar behavior of CRC-based, MD5-based and one-at-a-time implementations. Our SHA-based module had a significant less optimal performance than the other candidates.

B. Area Requirements of different signature implementations

In a resource-constrained environment hardware resource usage plays an important role, especially if the inclusion of a large amount of control signals should be enabled. Therefore,

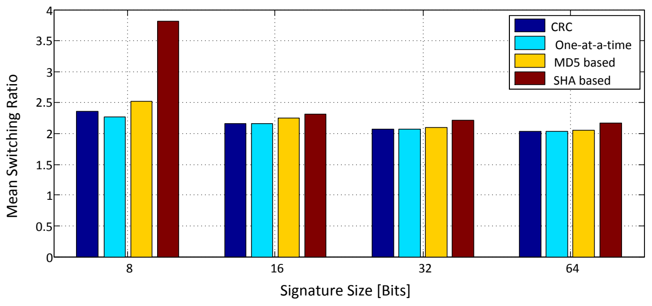


Fig. 6. Signature implementation input to output switching ratio

all modules have been synthesized for the Xilinx Virtex5 FPGA using different signature widths. Figure 7 shows the results of these synthesis runs concluding that all implementations scale as expected and one-at-a-time is needing the largest amount of FPGA slices. The lowest resource use was determined for all CRC-based modules.

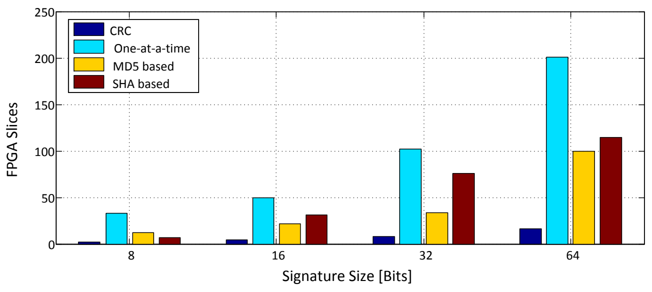


Fig. 7. Signature hardware resource usage

C. Evaluation using fault injection

After the signature hardware selection phase CRC proved to be the most promising candidate concerning hardware resource usage, scalability and hash performance. Therefore, a tree-structure has been implemented to generate a segmented hash containing information of three integrated processor cores. Each core outputs 63 state signals containing information about the processor's integer pipeline as well as register file, instruction cache, data cache, MMU and divider unit. This could be easily extended for further system units such as bus and network controllers. The monitored signals have been grouped into four segments per processor, resulting in five signature blocks per processor finally resulting in 15 hash generating hardware units. For small signature sizes (three, four, five) no compression has been implemented to avoid high collision ratios. The retrieved signatures contain 34 hash bits per processor core, resulting in a 102 bit system hash value.

Table I shows the resource usage of all signature generators combined in comparison to the complete system and one processor core. Furthermore, it provides a comparison to the implementations presented by Arora et. al. in [20]. Each of the monitored hardware units has been augmented using saboteur blocks to enable run-time fault injection.

TABLE I
RESOURCE USAGE COMPARISON

Unit	Slices	Slice Regs	LUTs	Resource [%]
System	15937	15017	30768	-
Core	3342	3248	6633	21.6
Signature Generator	103	102	263	0.85
Signature Controller	234	146	451	1.47
Arora et.al [20]	-	-	-	3-9[a]

[a]In [20] several different techniques have been proposed

To reduce the evaluation complexity only the first processor has been penetrated using 10000 randomly chosen fault patterns during our fault injection campaign. Table II gives an overview comparing our approach to the ones presented in [20] and [11].

TABLE II
FAULT DETECTION EFFICIENCY COMPARISON

Approach	Det. Bit Flips [%]	Det. latency [Instr.]
This work[a]	>99	1
Mao et.al Control flow [11]	26	23.6
Mao et.al Hash4 [11]	94	1
Arora et.al [20]	>99	6

[a]Single-round reduced CRC implementation

Our approach only needs a negligible amount of the system's resources while providing higher or similar detection rates than previously presented work. This result is achieved without influencing the execution performance as signature generation and comparison have been done in-system using dedicated memories. Communication to the integrity checking hardware is minimized to detection (de-)activation commands. Furthermore, in contrast to the earlier shown existing solutions our approach does not rely on any modification of the source or binary code.

VI. CONCLUSION

This paper presented a novel micro-architectural execution signature handling and characterization methodology for architectures providing existing control-state monitoring infrastructure. Our approach works fully transparent and does not rely on any changes of the executed software to detect unforeseen control flow changes. All presented hardware extensions have a low-impact on logic resources and can be efficiently integrated into existing power estimation infrastructure. Its modular design allows to generate execution signatures with varying signing granularity. Resulting signatures can then be used for transparent control-flow checking as well as for continuing modular-redundancy checking in high-security systems. The segmented approach allows for the direct localization of execution manipulations in a wide selection of system-on-chip submodules. All proposed circuits are fully synthesizable and have been successfully tested for their fault detection capability using an FPGA-based evaluation platform. Compared to existing state-of-the-art signature-based fault detection mechanisms our approach promises high detection efficiency without the need of additional watchdog hardware. The needed software overhead can be scaled according to the requirements of the chosen application.

ACKNOWLEDGMENT

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749. We would also like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

REFERENCES

[1] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-System Chip Multiprocessor Power Evaluations Using FPGA-Based Emulation," in *ISLPED*, 2008, pp. 335–340.
[2] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid, "Automated Power Characterization for Run-Time Power Emulation of SoC Designs," in *DSD*, 2010, pp. 587–594.
[3] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "A side channel attack countermeasure using system-on-chip power profile scrambling," in *IOLTS*. IEEE, 2011, pp. 222–227.
[4] M. Rebaudengo, S. Reorda, M. Torchiano, and M. Violante, "Soft-error detection through software fault-tolerance techniques," in *DFT*. IEEE, 1999, pp. 210–218.
[5] O. Goloubeva, M. Rebaudengo, M. Reorda, and M. Violante, "Soft-error detection using control flow assertions," 2003.
[6] R. Venkatasubramanian, J. Hayes, and B. Murray, "Low-cost on-line fault detection using control flow assertions," in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*. IEEE, 2003, pp. 137–143.
[7] N. Oh, P. Shirvani, and E. McCluskey, "Control-flow checking by software signatures," *Reliability, IEEE Transactions on*, vol. 51, no. 1, pp. 111–122, 2002.
[8] S. Daniels, "A concurrent test technique for standard microprocessors," *Dig. Papers Compcon Spring*, vol. 83, pp. 389–394, 1983.
[9] V. Iyengar and L. Kinney, "Concurrent fault detection in microprogrammed control units," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 810–821, 1985.
[10] K. Wilken and T. Kong, "Concurrent detection of software and hardware data-access faults," *Comp., IEEE Tr.*, vol. 46, no. 4, pp. 412–424, 1997.
[11] S. Mao and T. Wolf, "Hardware support for secure processing in embedded systems," in *DAC*. ACM, 2007, pp. 483–488.
[12] S. Lukovic, P. Pezzino, and L. Fiorin, "Stack Protection Unit as a step towards securing MPSoCs," in *IPDPSW*. IEEE, 2010, pp. 1–4.
[13] S. Kim and A. Somani, "On-line integrity monitoring of microprocessor control logic," *Microelectronics*, vol. 32, no. 12, pp. 999–1007, 2001.
[14] R. Ragel, S. Parameswaran, and S. Kia, "Micro embedded monitoring for security in application specific instruction-set processors," in *CASES*. ACM, 2005, pp. 304–314.
[15] Y. Fei and Z. Shi, "Microarchitectural support for program code integrity monitoring in application-specific instruction set processors," in *DATE*. EDA Consortium, 2007, pp. 815–820.
[16] K. Patel, S. Parameswaran, and R. Ragel, "Architectural frameworks for security and reliability of mpsocs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, no. 99, pp. 1–14, 2010.
[17] J. Smolens, B. Gold, J. Hoe, B. Falsafi, and K. Mai, "Detecting emerging wearout faults," in *SELSE*. Citeseer, 2007.
[18] J. Smolens, B. Gold, J. Kim, B. Falsafi, J. Hoe, and A. Nowatzyk, "Fingerprinting: bounding soft-error detection latency and bandwidth," in *ACM SIGARCH C.A.N.*, vol. 32, no. 5. ACM, 2004, pp. 224–234.
[19] A. Krieg, C. Bachmann, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "Accelerating early design phase differential power analysis using power emulation techniques," in *HOST*. IEEE, 2011, pp. 81–86.
[20] D. Arora, S. Ravi, A. Raghunathan, and N. Jha, "Secure embedded processing through hardware-assisted run-time monitoring," in *DATE*. IEEE Computer Society, 2005, pp. 178–183.
[21] B. Jenkins, "Hash functions for hash table lookup," Online, August 2011. [Online]. Available: http://www.burtleburtle.net/bob/hash/evahash.html
[22] Outputlogic.com, "CRC Generator," September 2011.

2012 15th Euromicro Conference on Digital System Design

PROCOMON - An Automatically Generated Predictive Control Signal Monitor

Armin Krieg, Johannes Grinschgl,
Norbert Druml, Christian Steger
and Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology, Austria
{armin.krieg, johannes.grinschgl,
norbert.druml, steger, rweiss}@tugraz.at

Holger Bock, Josef Haid
Design Center Graz
Infineon Technologies Austria AG
{josef.haid,
holger.bock}@infineon.com

***Abstract*—Today security and safety applications are often a large conglomerate of complex different components. Because of a strong trend to high system integration to fulfill financial and production cost constraints, as much of these components as possible are combined to form large-scale system-on-chips. Risks of dependability and security problems caused by device degradation and adversaries lead to a wide range of research concerning fault detection and recovery techniques in recent years. Especially in safety systems the concurrent use of different checking techniques, to protect the integrity of the operation, is preferred. Standard duplication or triplication methods for such critical devices are not completely fulfilling this property and raising a need for new on-line testing and recovery methodologies. Furthermore, the smart-card sector produces a strong need for new checking techniques with a low resource footprint. Therefore, this paper presents a novel automatized hardware generation flow to create a predictive control signal monitor unit in an automatized way. Depending on the instruction loaded by the processor pipeline this unit will predict the signature of following control signal changes. Hence, a new way of fault detection, weak checking, is implemented without introducing any large additional hardware blocks. A case study using an open-source processor is also presented to show the applicability of our approach.**

I. Introduction

The modeling and detection of faults has been a very active field of research in recent years. This work is motivated by the continuously intensifying problem of unreliable operation of highly integrated complex systems. This dependability issues are caused by sensitive, deep sub-micron structures used in current semiconductor manufacturing processes. Combined with high operating frequencies and low supply voltages the chance of random bit-flips is drastically increased. At the same time these complex embedded systems are introduced into an ever increasing amount of markets. In terms of necessary fault robustness two different types of fault sources have to be differentiated.

For high-availability systems like safety applications mainly random bit-flips are of concern to the engineer. Such faults can basically happen anywhere inside the implementation, but will only manifest themselves if the change has been stored in a register. Erroneous execution could lead to the activation of emergency units during normal operating resulting in catastrophic effects. Because of the risk to human life in many safety applications, there already exist strict standards requiring different control functionality to avoid false decisions.

In case of high-security applications the nature of the occurring faults is very different. The location and the time of activation is defined by the adversary that tries to put the system into an unintended state. The target of the designer of a secure implementation is not only to detect injected faults but also to prevent loss of trust and reduce available information to the attacker. The reaction time to fulfill this goals has to be kept as low as possible. A common solution to achieve fast fault detection is by modular duplication, drastically increasing the needed hardware resources. To make attacks even more difficult different control mechanisms are preferred, an aim that cannot be reached by replication of hardware blocks.

To combine the principle of different control mechanisms for the same target with high area efficiency and operating speed this work introduces a novel hardware generation flow for hardware checkers. This flow also includes new characterization steps to predict the control signal signature based on loaded instructions. The main contributions of this work are:

- The introduction of a generated signature-based memoryless execution monitoring unit.
- Presentation of a novel predictive control signal weak checking surveillance methodology.
- A case study using an open SoC implementation.

This paper is structured as follows. First, Section II gives a brief introduction into on-line monitoring techniques for processors and complex system-on-chips. Followed by Section III presenting a novel hardware monitor generation flow to provide area efficient and low latency fault detection. Section IV provides experimental results using an open-source system-on-chip platform to prove the applicability of our proposed approach. In Section V a summary of the current state-of-the-art concerning signature-based fault detection mechanisms is given. Finally, Section VI concludes this work with a short summing-up of the achieved results.

978-0-7695-4798-5/12 $26.00 © 2012 IEEE
DOI 10.1109/DSD.2012.106

II. On-line Monitoring of Processors and System-on-Chips

Fault detection and recovery have always been vital fields of research. Main reasons for this general interest have been influences by environmental stress like radiation and temperature. In recent years device degradation caused by highly integrated semiconductor manufacturing processes and intentional faults injected by an adversary moved into the focus of researchers.

A comprehensive overview over the current state-of-the-art in this field has been given in [1]. Many of previously proposed approaches to enable on-line detection in processor systems relied on high performance or area overheads. In this work we will consider techniques that can be used in resource-constraint systems without the need of implementing new system architectures at a slight cost of detection efficiency. For this work we reduce the abundance of different possibilities to implement such a fault-aware system to three major groups: *modular hardware redundancy*, *redundant execution* and *hardware checking*.

A. Modular hardware redundancy

The use of redundant hardware modules to enable fault detection capability is a classic approach to solve this problem. The reason for this is the simplicity of the system augmentation and implementation of the checking mechanisms. Depending on the used techniques performance overhead is usually very low but hardware cost is high. Therefore, only specific parts of the system are realized in a replicated way. Another important point of fault detection efficiency is detection latency. As the increased hardware cost is usually conflicting with cost constraints existing in many embedded applications, researchers tried to reduce duplication to sub-modules of the system at the cost of detection coverage. Such implementations concerning redundant or partitioned cores, pipelines, and ALUs have been shown in [2]–[4].

B. Redundant execution

Software based approaches usually take the way of redundant execution to enable fault robustness of a given design. This approach has been primarily used in safety systems because of the assumption that a random error will very unlikely happen twice at the same time and location. The main advantage of these solutions is the lack of necessary hardware changes. On the other hand detection latency is high and especially in slow applications time constraints prohibit the introduction of such measures. Another advantage compared to hardware redundant implementation is detection coverage, as the high level will cover every hardware module involved in the observed software algorithm.

C. Hardware checking

Similar to redundant hardware implementations, hardware checking also relies on the introduction of additional circuitry to allow for the testing of operation integrity. Classic approaches would be watchdog modules or build-in self-test (BIST) circuits checking certain assumptions about the system state. BIST-based and classic checker module-based solutions introduce performance overhead and high detection latencies depending on the implementation. Therefore, research in recent years concentrated on control flow-signature methodologies as shown in Section V. While these advances improved detection latency and hardware effort, there has been no comprehensive system observing approach usable in resource constraint systems. Hence, this work fills this gap in the field of fault detection implementations for such small embedded systems based on general-purpose architectures with a strictly defined structure.

D. Resource constraint systems

All these software and hardware-based techniques have significant disadvantages concerning resource-constraint systems. In case of hardware-based mechanisms, high detection accuracy is bought at the cost of high manufacturing costs. Software solutions have a strong impact on either execution performance or rely on large integrated memories to store execution signatures like hash values. While hash value generation for internal control flow properties has been shown to be a strong fault detection technique (as shown in Section V), hash generator hardware also results in a significant hardware demand. These algorithms are usually round-based to provide certain security properties necessary for their use in cryptographic applications. These round-based designs would result in long detection delays because the signature would only be valid after all rounds have been calculated.

Fortunately for pure fault detection purposes strict cryptographic properties do not have to hold, as the generated hash values are not used for critical calculations. Only the following characteristics have to be met by a signature algorithm used in pure on-line testing:

- *Collision robustness* : Especially if small hash signatures are used, collisions will happen, meaning that there are several input values with the same corresponding hash value. Therefore, a well designed fault detection signature hardware will be able to avoid such collisions even for low signature widths.
- *Switching ratio* : This ratio indicates how many bits of the output hash value will change compared to changing input bits. To retrieve highly unique signatures this ratio has to be as high as possible, a property that is also thought after in traditional hash algorithms.

Finally a signature generator implementation has to provide very low detection latency and low resource usage. Therefore, traditional hash and cyclic checking mechanisms can be stripped, meaning that they are reduced to only one generation round. Obviously, this reduced implementation still has to comply with the properties defined earlier. To find a suitable signature generation algorithm an additional characterization process is necessary (see [5]). For this work we build on previous work concerning the selection of a well designed hash generator and concentrate on techniques for an efficient, resource-saving memory-less monitoring system.

III. Novel Control Monitor Generation Flow

Recent years showed a variety of research concerning fault detection in high performance systems and safety applications. These domains, while suffering from the same lack of reliability and same security difficulties, are not strictly constraint in resource usage terms. Such highly constraint systems could be smart-cards, an application unifying high security demands with strong cost restrictions. Therefore, the introduction of software or hardware redundancy for fault detection purposes is often not a viable choice for the manufacturer. To fill this behavioral checking gap in the current state-of-the-art, a novel signature-based checking methodology has to fulfill the following properties:

- **Adaptability** : Usually on one basic design a variety of different applications are built upon. It is therefore important to support the automatized integration of a new checking infrastructure into existing source code.
- **Low resource usage** : Because of strict cost constraints, additional semiconductor area or memory space are to be kept as low as possible. Therefore, a memory-less pure logic implementation is preferable in such an environment.
- **Fast response** : Especially in the security domain fast reaction to intentional changes of the control flow are an important feature that has been neglected by work described in literature. Detection latency has to be kept below the pipeline length to prevent the writing of possibly secure data to an unsecured memory region.

To enable the fulfillment of all these properties a segmented approach is necessary, mainly to reduce necessary hash value widths but also to increase detection coverage. Therefore, we propose a *weak checking* methodology using several small monitors only covering a limited part of a program's temporal behavior. Combined a full temporal coverage can be achieved as depicted in Figure 1 for a given pipeline monitoring example.

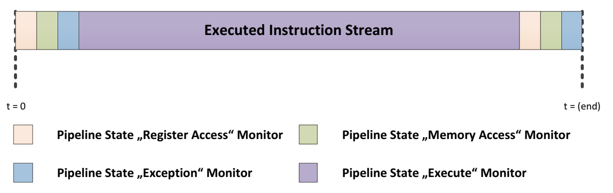

Executed Instruction Stream
t = 0
t = (end)
Pipeline State „Register Access" Monitor
Pipeline State „Memory Access" Monitor
Pipeline State „Exception" Monitor
Pipeline State „Execute" Monitor

Fig. 1. Execution monitoring coverage - Principle of *weak checking*, gain high detection coverage by several small monitors of low detection coverage

The following stages of the tracing and hardware generation process are based on existing pre-characterization tasks. These are necessary to determine relevant control signals of the target system elements. Such a signal-set could result out of a power-evaluation task or could be simply determined by an experienced system designer.

A. Code preparation

The augmentation of the existing hardware is not only necessary during the integration of the checker phase but also before characterization to enable fine grained tracing of internal processes. This preparation flow, as used in our proposed methodology, is shown in Figure 2.

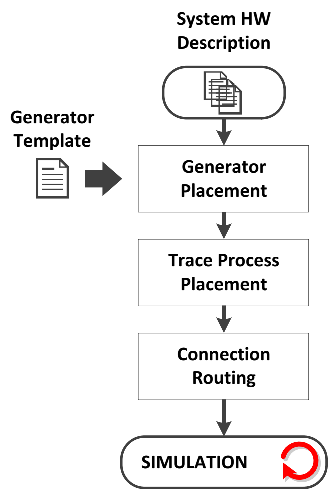


Fig. 2. System preparation flow - Automatized introduction of signature generator hardware and tracing processes for the efficient further evaluation of control signal behavior

- **Generator placement** : Depending on the chosen hash generation algorithm, appropriate generator modules are placed in the system-under-test. This process has to be done for every evaluated control signal which has been selected during a pre-characterization run.
- **Trace process placement** : For execution behavioral tracing purposes VHDL processes are placed into the target module to save all executed instructions and generated signatures.
- **Connection routing** : After all support modules and processes have been integrated into the existing hardware description, these have to be connected. This routing run finalizes the system for the characterization of the target application or benchmark suite.

After all these tasks have been completed, simulatable code for RTL simulation is available to be processed using any standard simulator like Mentor Graphics ModelSIM.

B. Execution characterization

To fulfill the *adaptability* property an automatized process for the generation of the checking hardware has to be implemented. Similar to control signal-based power estimation infrastructure generation (such an approach has been shown in [6]) standardized benchmarks can be used for such a characterization process. The approach of generating hardware by analyzing certain properties of the executed software opens also new possibilities for hardware/software pairing. The generic software characterization flow is shown in Figure 3.

- **Target application loading** : First the target application or part of a benchmark suite is loaded into the system-under-test.

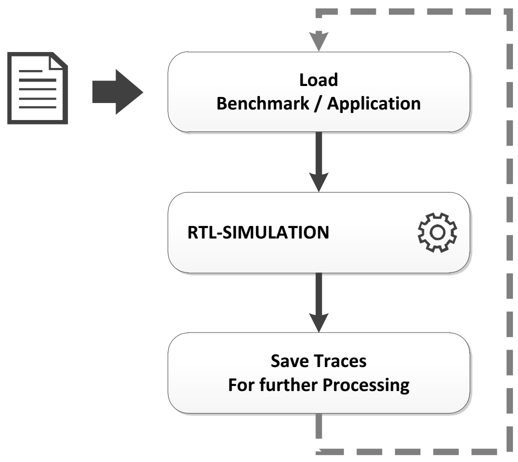


Fig. 3. Software characterization flow - RTL simulation based control signal behavior characterization using standardized benchmark suites or specific future application targets

- **RTL simulation** : For this characterization RTL simulation is sufficiently fast, so the application is executed using a common RTL simulator. The included tracing processes will now save all signal activity of selected control signals when activated by a software commando.
- **Trace storage** : All information gathered during this golden model run is now stored for further processing in MATLAB or a similar evaluation environment.

After characterization a small database of text-based trace files will be available to be analyzed during the following generation processes. Additional evaluation techniques could be introduced at this stage to improve detection coverage or gain information for fault or power modeling.

C. Hardware generation

After the control signal behavior of a selected application or a benchmark suite has been derived, this information can be used to generate the hardware checker modules. The basic generation flow, beginning from the relation extraction to the VHDL module implementation is depicted in Figure 4.

- **Characterization** : The characterization process itself is the largest part of the whole generation process, consisting of several sub-steps. First the execution traces from the golden model execution are loaded and parsed to extract only instructions and signatures recorded during the software part of interest (markers have been set in the application itself). Now all invalid data is removed, meaning multiple instruction-signature relationships or redundant information not usable during the further generation process.
- **Relation extraction** : After all invalid and unnecessary information were removed, the remaining instruction-signature pairs can be collected for further processing. To enable boolean minimization using tools like Espresso ([7]) these pairs are stored into a file with PLA-format.

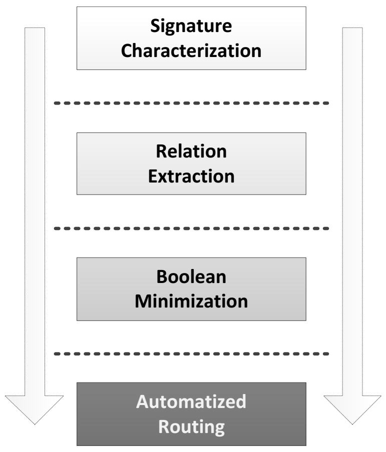


Fig. 4. Hardware generation flow - Dynamic generation of hash value predicting hardware based on an execution profiling characterization process

- **Minimization** : During this optional task a boolean minimization process can be applied to reduce logic complexity and further reduce resource usage.
- **Routing** : An automatized routing process is used to insert checker modules into existing source code and connect it to the control signals of interest. Depending on the chosen countermeasures in case of a detected fault, also result signals can be connected to this infrastructure.

D. Checking architecture

The hardware checking architecture itself is of a very simple structure, only containing a signature source and a monitor block checking these input values. Optionally an interface for evaluation can be included to check certain states of the current program execution. This basic architecture and the principle of its placement near to the processor's pipeline is depicted in Figure 5.

- **Signature generation** : Based on the current state of a selected amount of control signals, a representing signature is generated. The resource usage of the overall checking infrastructure is highly depending on the selection of efficient hash-generation hardware (as described in [5]).
- **Signature checking** : This process is mapped by the previous characterization and hardware generation processes into a static structure. If resource usage is not the uppermost concern during the design phase this can also be implemented using memories to store signature instruction pairs to increase on-line testing flexibility.
- **Result evaluation** : Optionally an evaluation and configuration interface can be provided to read out current checker states during execution or to reload internal memories in case of a flexible implementation.

This generic approach also allows to apply the same techniques to other parts of an integrated system, as long as there is a deterministic relation between a signature value and an observable state of the system.

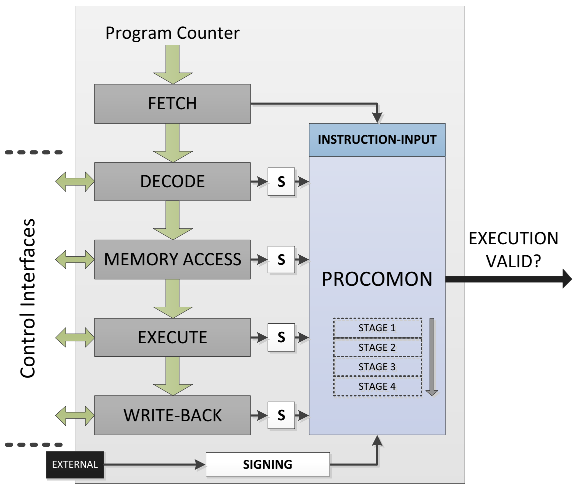


Fig. 5. Control signal based monitoring architecture - Specific implementation for a 7-stage pipeline in a SPARC v8 compatible CPU, each stage contains hash generators, PROCOMON predicts expected values depending on instruction

This specific example only provides on-line testing functionality for the processor's pipeline. That maybe sufficient if only simply control flow checking inside the CPU domain is demanded by the designer. If a more comprehensive approach is needed, a system-wide monitoring technique has to be used as suggested in Figure 6.

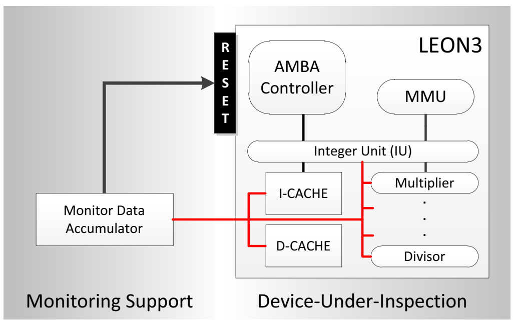


Fig. 6. Full system-on-chip monitoring architecture - An additional monitoring accumulator unit is responsible for the collection of eventual PROCOMON block alarms, activating for example a system reset in case of an emergency

IV. Experimental Results

To show the applicability of our chosen approach a widely known hardware platform based on an open source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler is used. It has been configured as shown in Table I and synthesized using Xilinx ISE software. All executed tests have been done using the ML507 evaluation board from the same manufacturer providing the synthesizer suite. The provided hardware description library includes not only a processor and memory environment but a large selection of peripherals that can be used as signature sources. MathWorks Matlab 2010b has been chosen to do all characterization operations using an six-core 3.2 GHz AMD Phenom-II machine equipped with sixteen giga-bytes of RAM.

TABLE I
LEON3 Processor Configuration

Operating Frequency [MHz]	40
Instruction Cache Sets	2
Instruction Cache Set Size [kB]	2
Data Cache Sets	1
Data Cache Set Size [kB]	2
MMU TLB Entries	8
MMU Page Size [kB]	prog.

The target of this experimental setup is to characterize parts of the MiBench benchmark suite. The results of this process are then used to generate signature monitoring units for the processor's pipeline. This augmented architecture is then thoroughly tested using a proven fault injection platform to derive the detection performance of the chosen implementation.

A. *Characterization results - CRC-based hash*

Depending on the target application investigated during the characterization the resulting checker complexity varies. This is a consequence of the fact that the current control signal state partially depends on the previous executed instructions. Multiple dependencies between instructions and signature have to be removed to provide unambiguous state detection.

TABLE II
Characterization Results - Instruction/Signature Pairs

Application	Reg-Pairs[b]	Exe-Pairs[b]	Mem-Pairs[b]	Xce-Pairs[b]
CoreMark	287	276	83	74
Dhrystone	100	142	80	64
basicmath[a]	145	344	141	141
bitcount[a]	87	64	92	98

[a]Round-reduced part of MiBench embedded systems benchmark suite
[b]Reg(register access), Exe(execution), Mem(Memory access) and Xce(exception) stage

These pairs can be evaluated in two ways, checking if the current signature for a certain instruction is correct or vice versa. The reason for this is that for certain execution profiles one or the other mode results in better characterization results. As this applies to the memory-access and exception-stages of the pipeline, they are implemented in swapped configuration. Meaning that the input of the PROCOMON-submodule will be the current stage-signature and the monitor will check if the current active instruction is valid.

B. *FPGA implementation results*

The final implementation of the pipeline checkers and an AMBA peripheral bus (APB) interface has been integrated into a LEON3 system-on-chip design. This implementation then has been synthesized and tested using our Xilinx Virtex5-based evaluation board.

TABLE III
RESOURCE USAGE COMPARISON

Unit	Slices	Slice Regs	LUTs	Resource [%]
System	7613	7413	15378	-
Core	3903	3567	7841	50.99
Signature Generators	28	109	102	0.66
PROCOMON[b]	407	0	1114	7.24
Arora et.al [8]	-	-	-	3-9[a]

[a]In [8] several different techniques have been proposed
[b]Depending on amount of signature-instruction pairs (bit_count)

Using standard settings provided by the LEON3 configuration the complete system has been implemented for our ML507 board. As shown in Table III our approach relies on less system resources than previously published work while providing comparable detection performance.

C. Fault injection efficiency

Based on a proven systematic fault injection approach (saboteur-based technique similar to the one presented in [9]) the proposed fault detection mechanisms is tested under faulty conditions. The results of these injection campaigns is compared to existing literature in Table IV.

TABLE IV
FAULT DETECTION EFFICIENCY COMPARISON

Approach	Det. Bit Flips [%]	Det. latency [Instr.]
This work	> 99	1
Mao et.al Control flow [10]	26	23.6
Mao et.al Hash4 [10]	94	1
Arora et.al [8]	>99	6

For these tests 10000 randomly injected faults into control signals of the pipeline have been used to determine the detection performance of our chosen approach. The results are compared to previous work published in [8] and [10].

V. RELATED WORK

State-of-the-art research in the field of fault detection mechanisms using on-line control signal monitoring can be divided into three main groups.

A. Software-based signature mechanisms

In case of purely software-based approaches automatic or semi-automatic generation of code signatures is used to detect program flow changes caused by tampering or environmental influences. For on-line testing these signatures are directly embedded into the executed binaries. The hardware signature checking procedure is itself relying on processor internal resources, resulting in high execution performance degradation.

In [11], a pure software solution using extensive redundancy has been proposed. An approach using additional state checking to detect unforeseen execution control behavior is shown in [12] and [13]. Software signatures, generated during compile time, are the base of the work presented in [14]. These software-only approaches have the distinctive disadvantage of resulting in a significant impact on execution performance.

B. Hardware-based signature mechanisms

Hardware monitors are the most commonly used hardware modules for integrity checking. The efficiency of these units is hereby directly dependent on the monitored system region. This advantage is lost if large signature storage memories are integrated to store precomputed signatures. The current state-of-the-art in this field fights another problematic situation, the selection of the monitored system region. Approaches only considering the processor pipeline may be insufficient for system-on-chips. The use of multiple monitors could be prohibitive in terms of system complexity and area efficiency.

Control flow-monitoring methodologies using dedicated monitoring hardware have been successfully introduced to reduce the impact of the on-line testing process on the execution performance. They also allow to gain direct access to the real-time behavior of hardware resources. Watchdog-type modules are classical examples for such dedicated monitoring hardware modules as shown in [15] and [16]. Smaller specialized monitoring circuits covering only selected parts of the system are presented in [10], [17], [18]. In [19], an approach relying on the modification of the processor pipeline to enable the observation of the control flow is presented.

C. Hardware-Software co-design solutions

Software/hardware co-design based approaches help to provide a wide detection coverage. The possibility to adapt both software support and hardware structure helps to reduce memory overhead and performance degradation. Because of the high grade of invasion into these systems application-specific instruction processors (ASIP) are the first choice for such implementations. Examples of such approaches have been published in several recent papers [20]–[22]. Of course, in many applications it is not possible or wanted that the design is build in such a way. For example if the targeted processor has been already thoroughly tested, recertification would lead to increased development costs. Therefore, in these cases these techniques cannot be applied.

Augmentation of existing hardware for the generation of representative values to track control changes inside the system-under-test would be another possibility. A proven approach for such reusable hardware blocks could be scan-out-chains methodologies as presented in [23]. However, this approach is not a real on-line testing solution as it relies on periodic tests that have a strong impact on execution performance. In [24], fingerprinting techniques generating hash-values from the complete architectural state have been published. In the resource-constraint systems domain strict constraints are defined concerning the resource-usage of checking approaches. In [5], different signature generation techniques have been characterized for signature size and detection efficiency.

D. Contributions

Our proposed approach can be categorized into group number three supporting system-internal fault detection mechanisms as well as software controlled solutions. Compared to previously published fault detection approaches we can provide very short reaction times because of the direct interpretation of the current system control state. The characterization-based monitor module allows to provide this functionality without relying on large memories of pre-recorded signature streams. The applied system execution characterization also provides for the absence of false positives during the evaluated time frame. Another advantage of our methodology is the lack of needed software augmentations and therefore, a complete lack of performance degradation. It is perfectly suited for resource constraint systems with small memories and tight cost constraints concerning development and production costs. Furthermore, the security and dependability of the complete system is not only based on a single type of fault detection, as it would be in a modular redundant implementation.

VI. Conclusion

In this work an automatized hardware generation methodology for on-line testing applications has been introduced. The generated signature-based processor and peripheral unit monitor allows for the immediate detection of unforeseen changes in the intended control flow. This approach does not rely on the augmentation of the executed software and even the signature characterization steps have been reduced into a single monitor generation process. Low detection latencies, low resource demands and the direct pairing of hardware and security-critical software especially fits this technique for the smart-card domain.

The applicability of our methodology has been shown using an open available system-on-chip implementation and the experimental results show a significant improvement compared to the current state-of-the-art in terms of detection speed, accuracy, and resource requirements. Detection efficiency has been evaluated using a proven fault injection emulation platform. Our future work includes further research concerning fault detection mechanisms with low resource requirements to replace modular redundancy approaches in smart-card and general system-on-chip implementations.

Acknowledgment

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749. We would also like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

References

[1] D. Gizopoulos, M. Psarakis, S. Adve, P. Ramachandran, S. Hari, D. Sorin, A. Meixner, A. Biswas, and X. Vera, "Architectures for online error detection and recovery in multicore processors," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2011, pp. 1–6.

[2] S. Hong and S. Kim, "Lizard: Energy-efficient hard fault detection, diagnosis and isolation in the ALU," in *Computer Design (ICCD), IEEE International Conference on*. IEEE, 2010, pp. 342–349.

[3] R. Vadlamani, J. Zhao, W. Burleson, and R. Tessier, "Multicore soft error rate stabilization using adaptive dual modular redundancy," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2010, pp. 27–32.

[4] J. Yao, R. Watanabe, K. Yoshimura, T. Nakada, H. Shimada, and Y. Nakashima, "An Efficient and Reliable 1.5-way Processor by Fusion of Space and Time Redundancies," 2011.

[5] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, A. Genser, H. Bock, and J. Haid, "Characterization and Handling of Low-Cost Micro-Architectural Signatures in MPSoCs," in *IEEE European Test Symposium (ETS)*. IEEE, 2012, pp. 62–67.

[6] A. Krieg, C. Bachmann, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "Accelerating early design phase differential power analysis using power emulation techniques," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE Symposium on*. IEEE, 2011, pp. 81–86.

[7] M. Ciesielski, S. Yang, and M. Perkowski, "Multiple-valued Boolean minimization based on graph coloring," in *Computer Design: VLSI in Computers and Processors, 1989. ICCD'89. Proceedings., 1989 IEEE International Conference on*. IEEE, 1989, pp. 262–265.

[8] D. Arora, S. Ravi, A. Raghunathan, and N. Jha, "Secure embedded processing through hardware-assisted run-time monitoring," in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*. IEEE Computer Society, 2005, pp. 178–183.

[9] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock, and J. Haid, "Modular fault injector for multiple fault dependability and security evaluations," in *Digital System Design (DSD), 2011 14th Euromicro Conference on*. IEEE, 2011, pp. 550–557.

[10] S. Mao and T. Wolf, "Hardware support for secure processing in embedded systems," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 483–488.

[11] M. Rebaudengo, S. Reorda, M. Torchiano, and M. Violante, "Soft-error detection through software fault-tolerance techniques," in *Defect and Fault Tolerance in VLSI Systems, International Symposium on*. IEEE, 1999, pp. 210–218.

[12] O. Goloubeva, M. Rebaudengo, M. Reorda, and M. Violante, "Soft-error detection using control flow assertions," 2003.

[13] R. Venkatasubramanian, J. Hayes, and B. Murray, "Low-cost on-line fault detection using control flow assertions," in *On-Line Testing Symposium, 9th IEEE*. IEEE, 2003, pp. 137–143.

[14] N. Oh, P. Shirvani, and E. McCluskey, "Control-flow checking by software signatures," *Reliability, IEEE Transactions on*, vol. 51, no. 1, pp. 111–122, 2002.

[15] S. Daniels, "A concurrent test technique for standard microprocessors," *Dig. Papers Compcon Spring*, vol. 83, pp. 389–394, 1983.

[16] V. Iyengar and L. Kinney, "Concurrent fault detection in microprogrammed control units," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 810–821, 1985.

[17] K. Wilken and T. Kong, "Concurrent detection of software and hardware data-access faults," *Computers, IEEE Transactions on*, vol. 46, no. 4, pp. 412–424, 1997.

[18] S. Lukovic, P. Pezzino, and L. Fiorin, "Stack Protection Unit as a step towards securing MPSoCs," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–4.

[19] S. Kim and A. Somani, "On-line integrity monitoring of microprocessor control logic," *Microelectronics journal*, vol. 32, no. 12, pp. 999–1007, 2001.

[20] R. Ragel, S. Parameswaran, and S. Kia, "Micro embedded monitoring for security in application specific instruction-set processors," in *Proceedings of the 2005 CASES*. ACM, 2005, pp. 304–314.

[21] Y. Fei and Z. Shi, "Microarchitectural support for program code integrity monitoring in application-specific instruction set processors," in *Proceedings of the conference on Design, automation and test in Europe*. EDA Consortium, 2007, pp. 815–820.

[22] K. Patel, S. Parameswaran, and R. Ragel, "Architectural Frameworks for Security and Reliability of MPSoCs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, no. 99, pp. 1–14, 2010.

[23] J. Smolens, B. Gold, J. Hoe, B. Falsafi, and K. Mai, "Detecting emerging wearout faults," in *Third IEEE Workshop on Silicon Errors in Logic–System Effects (SELSE 3)*. Citeseer, 2007.

[24] J. Smolens, B. Gold, J. Kim, B. Falsafi, J. Hoe, and A. Nowatzyk, "Fingerprinting: bounding soft-error detection latency and bandwidth," in *ACM SIGARCH Computer Architecture News*, vol. 32, no. 5. ACM, 2004, pp. 224–234.

Power And Fault Emulation for Software Verification and System Stability Testing in Safety Critical Environments

Armin Krieg, *Student Member, IEEE*, Christopher Preschern, Johannes Grinschgl, *Student Member, IEEE*, Christian Steger, *Member, IEEE*, Christian Kreiner, *Member, IEEE*, Reinhold Weiss, *Member, IEEE*, Holger Bock, and Josef Haid

***Abstract*—In recent years the complexity of digital control systems in safety critical environments increased steadily from simple discrete control units to complex embedded systems. A wide industrial consensus about the necessity of a set of safety definitions lead to the introduction of several functional safety standards like IEC61508. To achieve that novel embedded systems comply with these requirements, thorough testing is needed during early design stages of the integrated device. Currently only fault injection testing using manufactured products and netlists of system-on-chips are used to determine the fault resistance of the embedded system. This late testing could result in expensive redesigns and hide implementation errors because of the black-box approach. This approach is also not practicable if software and hardware providers are separate entities. This paper presents a flexible fault injection and power estimation platform to enable thorough examinations of novel complex system-on-chips for automotive or similar critical environments. The microprocessor evaluation approach is extended with smart bus fault emulation units for common buses like Ethernet. The combined power and fault emulation techniques allow for the instant exploration of eventual power supply peaks and implementation weaknesses.**

***Index Terms*—Automotive embedded system, fault injection, fault tolerance, power estimation, software verification.**

I. Introduction

The safety system-on-chip manufacturing markets, like the automotive industry, have seen a tremendous increase of system complexity in recent years. Main drivers behind this development are not only more passenger comfort functions, but also more complex engine management, high amounts of smart sensors and finally a growing number of safety functionalities. As these systems often consist of a large amount of different individual units, large bus-systems are necessary to connect this functionality. The control system manufacturer and specifically semiconductor design centers have an increasing problem concerning the test of this mixed software/hardware implementations. With the introduction of FPGAs into critical safety-relevant control systems, additional reliability problems of these devices have to be considered [1], [2]. Finally, new safety standards like IEC 61508 for systems in critical environments like automotive or energy production applications promote extensive testing and simulation to provide correct operation.

In recent years two main strategies have been selected to ensure a final device is tested for common sources of operational faults. First, model-based approaches help to identify design deficits at a very early stage. This approach is highly dependent on very accurate system models and some implementation characteristics are not available until the system has been manufactured. The second approach would be to test the final device inside an environment that closely resembles the final target environment. These tests can only be done at a very late stage when a final device is available. Some information like the power consumption behavior are also of high interest to the software and system developers [3].

Existing work trying to fill this gap inside the design test chain is only considering very limited cases of random single event upsets. Research has shown that such simple fault models are not sufficient to model faults resulting of degradation in devices manufactured using deep submicron semiconductor manufacturing processes [4]. This is especially true for recent safety standards concerning multi-bit-upsets, which also rely on spatial proximity information [5]. In modern safety control systems not only safety but also security concerns have to be considered as critical systems are indirectly connected to entertainment modules. Therefore, this work provides a comprehensive platform to enable the designer of safe system-on-chips to not only evaluate software and hardware for their fault resistance but also for its power consumption behavior. The main contributions of this work are

- Extension of power and fault emulation approaches into the safety system engineering domain under consideration of standards such as IEC 61508 or ISO 26262.
- Introduction of a novel high-level multi-disciplinary processor and bus-system testing methodology.
- A case study using an open-source implementation of a SPARC-based processor including bus modules.

Manuscript received April 15, 2012; revised July 30, 2012; accepted September 17, 2012. This work was supported by the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes project under FIT-IT contract FFG 825749. Paper no. TII-12-0283.

A. Krieg, C. Preschern, J. Grinschgl, C. Steger, C. Kreiner, and R. Weiss are with the Institute of Technical Informatics, Graz University of Technology, Graz 8010, Austria (e-mail: armin.krieg@tugraz.at; christopher.preschern@tugraz.at; johannes.grinschgl@tugraz.at; christian.kreiner@tugraz.at; steger@tugraz.at; rweiss@tugraz.at).

H. Bock and J. Haid are with the Design Center Graz, Infineon Austria AG, Graz 8010, Austria (e-mail: holger.bock@infineon.com; josef.haid@infineon.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2012.2220976

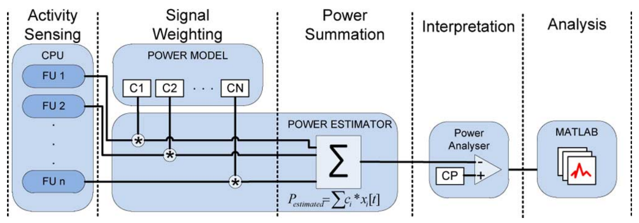


Fig. 1. Power emulation principle—Control activity of selected functional units (FU) is transformed to cycle accurate power estimates (adapted from [7]).

This paper is structured as follows. First, Section II gives a brief introduction into principle power and fault emulation techniques. Second, Section III summarizes the current state-of-the-art concerning reconfigurable software and hardware testing systems. Followed by Section IV presenting a novel FPGA-based evaluation platform for combined software and hardware testing in safety applications. In Section V experimental results using an open available system-on-chip implementation are shown to prove the applicability of this work. Finally, Section VI gives some conclusive words describing the impact of this paper.

II. Power and Fault Emulation

A. Power Consumption Emulation

The principle of *power emulation* (PE) as a state-of-the-art functional emulation using hardware-implemented power models has been introduced by Coburn *et al* in [6]. Meaning that the functional emulation and power estimation process are executed concurrently during the run-time of the system.

Depending on the application field the generated approximated power values can be immediately used for further processing or saved as traces in case of power consumption evaluations, as depicted in Fig. 1.

1) Extended Power Emulation Unit: Power consumption estimations are provided using a high-level *power emulation unit* similar to those shown in [8]. As depicted in Fig. 1 it constitutes a direct implementation of the power macro model approach.

The operation of this module is based on additive linear macro models to generate static and dynamic power consumption estimates (as expressed in (1)). Monitoring of power-relevant control signals x_i results into estimates for several different system components. These components are weighted using the values c_i from the hardware integrated power macro model. The static part of the power consumption is described using the coefficient c_0. Both, x_i and c_i are determined through a thorough benchmark-based characterization process as proposed in [8].

$$\hat{P}[t] = \hat{P}_{sta} + \hat{P}_{dyn}[t] = c_0 + \sum_{i=1}^{N} c_i x_i[t]. \tag{1}$$

Interference-free run-time power tracing is enabled by extending the traditional emulation platform with an internal and configurable FIFO buffer. This memory decouples the storage of the trace from the evaluated control algorithms to remove eventual profile distortions. After each evaluation process its contents are safely copied to an external memory device.

B. Fault Injection Testing

Fault injection means the intentional introduction of faults into a system for the simulation and emulation of errors. Such operation disruptions can be caused by external influences like radiation, attacks or internal reasons like device degradation. Depending on the level of abstraction, several different methods have been described in literature:

- **Hardware-Level**: At this abstraction level manufactured devices are available and existing automated test equipment can be reused. By nature the possible points of fault injection are very limited. Such points can be physical in- or outputs (pin-level) which can be used to manipulate data and control flow. Internal error sources can be simulated through external radiation sources or directly by physical manipulation of the silicon device.
- **Software-Level**: The basic principle behind software fault injection is the same as in hardware FI. The main difference lies in the manipulation target, variables and other system elements are changed directly while the hardware works as originally designed.
- **Modeling-Level**: Especially for designers of embedded systems fault testing during earlier development stages, before a final device is available, is of utter importance. Therefore several different techniques using direct manipulation of the RTL description, partial run-time reconfiguration of FPGA resources or by utilizing simulator commands have been shown in the past.

To provide early access to software power consumption data and fault resistance results, our approach is based on the automatized augmentation of the hardware description (similar to [9]). Basically there are two possible ways to achieve evaluation results on RTL level, one would be by using cycle accurate simulation and the other by executing the adapted hardware model on a reconfigurable test platform.

III. Related Work

The reliability of systems used in space flight applications has been of concern for many years leading to a wide range of publications about test methods using fault injection techniques. The target of such fault injection campaigns can be completely manufactured parts for example using radiation or if early tests are needed during the design phase using manipulation of the hardware description.

The advantage of high-level hardware descriptions to be simulatible directly during the design phase was exploited in early fault injection tools based on simulation techniques. One of these first simulation tools was MEFISTO specifically targeted on fault injection into VHDL models [10]. A pure simulation-based technique has been complemented with automatized saboteur and mutant insertion strategies for the VFIT tool in [11]. In [12] further improvements concerning the injection performance of simulation and emulation approaches have been presented. Similar to these previous works and our chosen concepts, flexible automatized routing approaches have been shown in [9].

The need for higher fault injection coverage led to the heavy research activity in the field of emulation techniques. As shown in [13] emulation promises significantly higher injection rates and therefore enables the possibility to evaluate more possible fault configurations than using simulation. In [14] novel partial reconfiguration capabilities of certain new FPGAs are used to enable fault injection without modification of the system code base. Further improvements concerning performance and practicality have been introduced in [15], [16]. The work presented in [5] additionally enables the importing of netlists into the FPGA-based system and is using proximity information for correct and fast MBU robustness investigations.

The main motivation behind these approaches has been dependability evaluations. Security evaluations by nature (intentional faults could happen at several positions at once) result in more complex fault scenarios therefore multiple fault models have to be considered as suggested in [17]. This is now also valid for many highly integrated systems not destined to the security domain because of reliability issues of deep-submicron semiconductor process technologies.

Specifically in the automotive industries emulation-based system fault testing is already playing a vital role as shown in [18]. Their approach is based on gate-level fault emulation, a very low abstraction often unwanted for software evaluations during early design stages. Systematic high-level methodologies for the testing of automotive communication systems are shown in [19], [20]. Specifically targeting a CPLD-based fail-safe system the authors of [21] applied a saboteur-based approach. In this case tests are still done at a very low system level to ensure correct function of this safety implementation. To increase emulation capacity and performance a parallelized approach is presented in [16]. Still the authors only consider random fault distributions and single hardware modules. Therefore, it is of vital importance to create a bridge between low-level hardware verification and high-level system verification.

Hence, a hybrid platform combining the advantage of state-of-art verification systems as shown in [22] and very low-level emulation approaches as presented in [23] is needed. The importance of accelerated evaluation techniques for software verification in the security domain has been shown in [8], but this work considered information leakage as its primary concern. To the best of our knowledge there is currently no such comprehensive approach, also considering the dynamic power consumption behavior, available. Also, there is no flexible fault injection approach for the early verification of high-level software safety and security specifications, which clearly specify high-level fault effects.

IV. Power and Fault Emulation-Based Safety Test Methodology

Large scale integration resulted in tremendous problems concerning power consumption and fault resistance. To counteract possible false behavior after market introduction a comprehensive methodology is needed. Some of the necessary infrastructure tasks to enable power and fault emulation are very similar, hence they can be combined into a single FPGA-based platform.

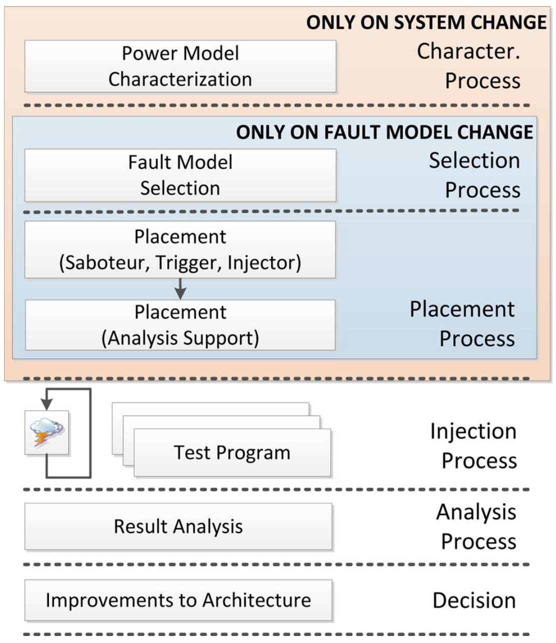


Fig. 2. Hardware accelerated fault emulation flow—From system characterization to final architecture and software optimization decisions.

The emulation flow starting with the model power characterization and ending with architecture improvements is depicted in Fig. 2. This emulation flow consists of six principle stages, three of which only have to be done after significant changes to the target system.

- **Power model characterization**: This time-consuming power simulation based process has only to be executed if significant changes to the modeled system or units have been done.
- **Fault model selection**: Especially for safety systems special care has to be taken when choosing an appropriate fault model. The aimed Safety Integrity Level (SIL) and the application of the chip should already be known at this point. Depending on the desired SIL level and application (e.g., processing unit or discrete digital circuit) different fault models have to be considered for the simulation or emulation. Most of the fault models consider single stuck-at or transient faults and focus on memory registers.
- **Placement process**: Only if the power characterization or the fault model change, new analysis hardware has to be placed into the evaluation platform.
- **Injection Process**: During the injection process a pre-tested workload is loaded into the system-under-test. Also the emulation controller is configured according to the chosen fault models and investigation targets.
- **Analysis process**: The injection processes result in a significant amount of data that has to be processed in order to make a sound decision.

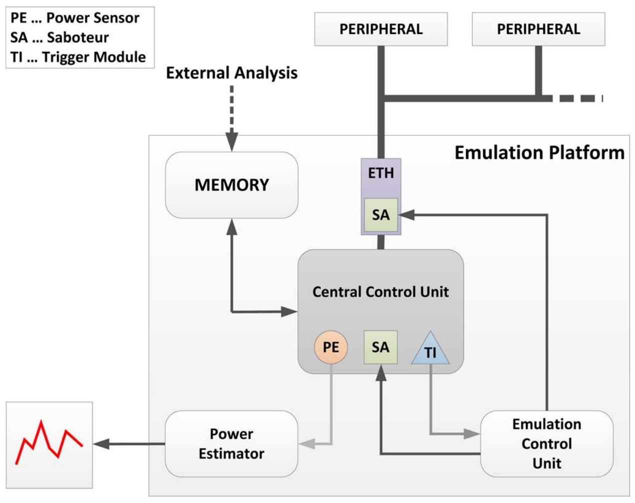


Fig. 3. FPGA-based system test platform.

The integrated power consumption and fault injection architecture is depicted in Fig. 3. Using an automated VHDL hardware description augmentation approach five different types of modules and signals are routed into the system-under-test.

- **Power estimator**: Top level module collecting all system state signals to generate a power estimate.
- **Fault emulation control unit**: Top level unit to provide an automated emulation flow.
- **Power sensors**: Signals to power consumption relevant signals are connected to the power estimator.
- **Trigger modules**: Small trigger modules to allow the observation of fault activation conditions.
- **Saboteurs**: Small signal disturbing units to emulate the low-level fault behavior.

Depending on the verification process, trigger modules and power sensors are optional. Randomized testing as used in hardware verification will prefer fast injections without the consideration of architectural hardware elements to more targeted approaches as used in software verification.

A. Fault Injection Implementation

State-of-the-art fault injection implementations for safety applications rely on statistical methods for randomly activated faults. This approach results into several problems to the designer of the evaluation platform and the system design engineer.

- Very high fault injection speeds are necessary to provide enough data for statistical fault injection investigations.
- Analysis of the results of such a campaign are difficult to interpret, meaning problematic hardware descriptions are difficult to identify if a certain fault injection campaign resulted into an error.
- While hardware verification relies on test coverage completeness of the final gate-level design, high-level software verification relies on a more abstract view. Results of netlist fault injection campaigns are only useful when lower-level hardening and replication is applied [24].
- If security or safety concerns play a role during the investigations, random injections are not sufficient anymore. In both cases the parts of a system-on-chip contain specific countermeasures against certain pre-defined fault scenarios. This results out of the abstracted view an adversary or software developer has of the attacked/investigated system. These countermeasures have to be accurately tested to achieve correct functionality during the scenario they have been designed for.

To enable efficient emulation that can satisfy both, hardware and software verification needs, a saboteur-based approach has been chosen. This is possible because in this work only high-level fault detection mechanisms targeting high-level fault effects (i.e., at address 0x100 and cycle 1000 bit 3 of word 0 flips) are concerned. Additional evaluation flexibility is gained by a modular trigger concept, allowing for the use of basically every hardware signal as a fault activation source.

1) Safety Standard Considerations: To conquer the problematic implications of the introduction of complex distributed systems into several critical applications, several functional safety standards have been defined. In the safety critical system engineering industry sector two are of outstanding importance:

- **IEC 61508**: A general market consensus standardized by the international electro-technical commission (IEC)[25].
- **ISO 26262**: A specific functional safety standard for the automotive industry, standardized by the international organization for standardization (ISO) [26].

Both standards include the definition of *Safety Integrity Levels—SIL* metrics for hardware safety integrity and hardware fault tolerance (ASIL in case of ISO 26262). The following failure rate metrics have been defined by IEC 61508:

- λ_S: rate of failures that do NOT affect system safety;
- λ_{SD}: detected safe failure rate;
- λ_{SU}: undetected safe failure rate;
- λ_D: rate of failures that result in safety degradation;
- λ_{DD}: detected dangerous failure rate;
- λ_{DU}: undetected dangerous failure rate.

For system level safety the so called *Safe Failure Fraction* metric has been defined. This function is defined as shown in (2). This ratio of safe and detected dangerous failures to the complete failure rate is used to determine the SIL level of the evaluated system

$$SFF = \frac{(\lambda_{DD} + \lambda_{SD} + \lambda_{SU})}{(\lambda_{DD} + \lambda_{DU} + \lambda_{SD} + \lambda_{SU})} = \frac{\lambda_S}{\lambda_D + \lambda_S}. \quad (2)$$

For the calculation of the SFF metric the lambda values have to be known for all sub-components. These λ-values are either provided by the manufacturer of a certain system element, or are determined during the FMDEA (Failure Mode Diagnostic and Effects Analysis) process. Alternatively these values can be derived using system simulation or emulation techniques. Therefore, it is of importance to allow for the early evaluation of customized integrated circuits. Another metric important in this context is *Hardware fault tolerance (HFT)* which is defined in the following way:" HFT N = N + 1 faults could cause a loss of the safety function." Especially for the HFT analysis the consideration of multiple-bit faults is important. The usage of classic full coverage hardware verification tools is not feasible in case of industrial settings which treat gate-level netlists as a company secret. Therefore, a targeted approach using high-level fault models is needed to enable external entities to test their software or hardware implementations for fault robustness.

Depending on the SIL level that the designers of a novel system aim at, the safety standards demand the consideration of design-for-testability and ”resilient functioning” techniques during design time. This resilient implementation has to be verified by fault injection methodologies as required by the standard for ASIC and FPGA devices.

B. Evaluation Methodology

The finally applied evaluation methodology highly depends on the group of engineers using the proposed approach for their system investigations. The targets of software and hardware verification engineers are very different, therefore both verification techniques have to be considered for their own. An even higher-level view is taken by engineers testing inter-system communication networks, mainly considering protocol measures to ensure communication integrity.

1) Hardware Verification: Hardware verification is relying on very high test coverage of the hardware implementation, hence random testing using widely spread saboteurs is the first choice. To be able to analyze the high amounts of data resulting of such random fault injection campaigns, already partitioning is used to test submodules inside the complete system environment.

This approach is directly supported by the chosen automatized hardware description augmentation approach while strictly random injection implementations need deeper intrusions into the system design. While it is possible to augment the system with a large number of saboteurs, it is more suited for the testing of hardware interfaces inside of system-on-chips. This way the evaluation complexity is reduced, while still enabling the verification of hardware interface specifications.

Therefore, a practical approach includes the selection of a unit-under-test, augmentation of the hardware description of this unit and finally long-time testing of this unit. The FPGA-based approach allows for the parallel testing of several units at the same time in real-time, allowing for a quick and thorough investigation of the complete system-on-chip. During these campaigns power consumption traces can be stored for further analysis to avoid eventual power bugs that have not been detected during traditional power simulations.

2) Software Verification: The software design and verification engineers consider a very abstracted view on the used system. Many hardware parts are hidden by architectural elements like memories, registers and program counters. Random testing on the flip-flop or even transistor-level is therefore unpractical to ensure the functionality of software fault detection and recovery implementations. This is especially problematic as some safety relevant functions are implemented in software to avoid expensive redesigns of the hardware architecture (diversified programming as considered in IEC 61508–3).

Therefore, a modular trigger-system has been implemented to observe such architectural elements and allow for the random injection of faults into memories, registers etc. during predefined time frames inside the software implementation. In this case the hardware description is only modified once to place trigger and saboteur modules. Further test configurations are done using the configuration of the fault injection controller. As for the hardware verification, power profile tracing is enabled through all injection campaigns to ensure a power-aware software implementation.

3) Communication Testing: Reliable communication protocols feature several countermeasures to allow for the correct transmission even under environmental stress. Usually error correction capabilities are tested by sending data packages that violate certain protocol assumptions. This can be done using an external tester system or using a saboteur at the bus peripheral or controller units. Like in the software verification methodology smart trigger modules can be used to specifically target certain bits of the current transmission. Our saboteur approach allows not only for the emulation of bit-flips and stuck-at faults (not useful for communication testing) but also of certain delay effects as they exist in realistic implementations. It therefore fulfills the requirement of IEC 61508 to test a safe communication protocol for its ability to detect lost or corrupted packages.

V. Experimental Results

A generally available system-on-chip platform based on an open source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler is used to show the applicability of our approach. An exemplary configuration using Ethernet bus peripherals is chosen and has been synthesized using Xilinx ISE software. The result has been tested on the ML507 evaluation board manufactured by Xilinx. While this processor configuration cannot be classified as a typical example of a safety system implementation, it is still possible to prove the basic principles of our methodology as it can be used for any application available in form of RTL source code. As an emulation control unit, the PowerPC 440 processor integrated into the Xilinx Virtex5FXT FPGA has been chosen to achieve independent operation from the evaluated target system. Analytical results and characterization data is produced using MathWorks Matlab 2010 b on an six-core 3.2 GHz AMD Phenom-II machine using sixteen giga-bytes of RAM. All evaluated applications have been taken from the widely known and heavily tested MiBench benchmark suite [27].

After system characterization a power macro model containing 63 coefficients using the semi-custom design flow and power simulation tools provided by our industrial partner has been derived. This large macro model has been implemented inside the power estimation module using a three stage pipelined adder structure.

A. Automotive Software Power Characterization

Power consumption characterization of a software implementation is important because of several reasons. If the targeted system only provides a very limited power budget, consumption peaks could lead to a fluctuations in the power supply. This will lead to the disruption of the software execution or even other devices supplied by the same supply [28]. In case of secure applications an adversary could retrieve information about the implementation usable for a possible attack.

To prove the applicability of our power emulation approach Fig. 4 shows the power consumption profile of the MiBench

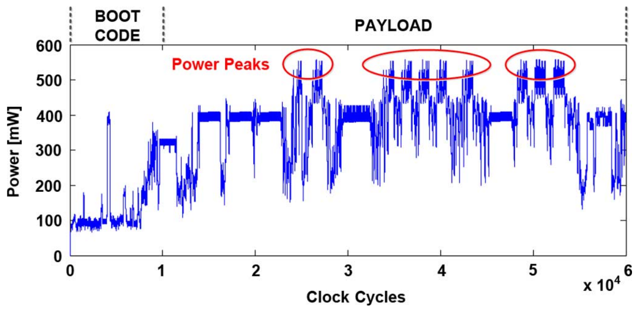


Fig. 4. Power consumption profile of benchmark basicmath_small.

automotive benchmark *basicmath_small*. While a simple embedded processor will not have a strong influence on wire-powered system, it is still a good example to demonstrate the software power verification process.

In Fig. 4 several distinctive consumption peaks can be identified. If this current implementation violates certain power supply constraints the developer effectively has two choices:

- Change the implementation: Switching of idle system units, usage of power saving processor instructions or reducing the ”consumption density”. The latter means the use of a slower but less aggressive implementation.
- Dynamic voltage and frequency scaling (DVFS): Trying to find a suitable trade-off between high operating frequency and high supply voltage for high execution speeds and high power consumption efficiency.

1) Experimental Conclusion: Based on an open available automotive application selection, a typical domain dependent software load has been characterized. This standard implementation showed several power consumption peaks during its operation that could lead to potential reliability problems. This detailed temporal power consumption information can be directly used to optimize application behavior for better resource usage.

B. Fault Resistance Evaluation

To demonstrate the variability of our saboteur-based fault injection methodology three evaluation targets of our system-on-chip have been chosen. First, the implementation of a simple client-server communication protocol is tested by placing saboteurs between the Ethernet MAC-controller and its physical interface. Second, the effects of high environmental stress on the FIFO-memory of a peripheral controller are emulated. Finally, the software implementation of typical automotive workload as provided by the MiBench benchmark suite is tested for its robustness against multiple bit-flips in its main memory. As a low-level fault model stuck-at-one and stuck-at-zero models have been chosen, but also the change to a bit-flip model would only require a short reconfiguration of the saboteurs.

1) Ethernet Communication Testing: A typical client-sever communication protocol has been implemented to provide a defined transmission and reception process between the Ethernet MAC-controller and its physical interface. The embedded system-on-chip has been configured as the data producing client (running MiBench’s *basicmath_small* benchmark) and a standard PC is always resending the same package back to the client. Two basic mechanisms have been implemented to provide the detection of lost or corrupted data packages. Data corruption is detected by a CRC8 and CRC16 checksum and data loss is identified by a sequence number integrated into every data packet.

TABLE I
FAULT INJECTION INTO ETHERNET COMM. RESULTS

Benchmark	Inj. Faults	CRC Fails	Sent Pack.	Receiv. Pack.	Lost [%]
Basicmath_small (Client)	>255000	0	13860	9761	29.6
Basicmath_small (Server)	>255000	0	9761	9761	-
Basicmath_large (Client)	>162000	0	1900	302	84.1
Basicmath_large (Server)	>162000	0	302	302	-

TABLE II
FAULT INJECTION INTO ETHERNET CONTROLLER MEMORY RESULTS.

Benchmark	Inj. Faults	CRC Fails	Sent Pack.	Receiv. Pack.	Lost [%]
Basicmath_small (Client)	> 260000	10	7929	7019	11.5
Basicmath_small (Server)	> 260000	835	7094	7929	-
Basicmath_large (Client)	> 248000	3	1608	1361	15.4
Basicmath_large (Server)	> 248000	157	1451	1608	-

The result of these long-time fault injection campaigns are shown in Table I. Both benchmark programs have very different execution profiles, basicmath_small needs less computations resulting in a lower turn-around time. The communication between MAC-controller and physical interface is using fault detection, hence, no data corruption but only lost packages can seen by the communication software.

2) Peripheral Controller Testing: In this test case typical random faults resulting of environmental stress or device degradation is emulated. Faults are randomly injected into the FIFO memories of the Ethernet controller. The same communication protocol as during the basic communication tests has been chosen including the same software fault detection mechanisms.

As can be seen in Table II the hardware fault detection mechanisms are not able to detect latent memory data faults. Therefore, data corruption can be seen in both, received and sent data packages. All data corruption has been caught using the implemented software fault detection mechanisms.

3) Embedded Controller Software Verification: Specifically in implementations using software based system integrity checks it is vital for functional safety to obtain information about the diagnostic coverage (lamda values) of these tests. The automotive applications from the MiBench benchmark suite have been augmented with simple memory integrity checks and its efficiency is tested using fault injection into the cache bus. A set of critical variables of the application program has been defined to reduce the fault injection target memory space. Trigger-based fault injection enables the testing of fault detection software parts in a realistic operating system environment without influencing the OS itself.

The temporal redundant calculation implementation detected all calculation errors, at the cost of a doubled execution time.

Depending on an eventual hardware error detection, such faults at the cache-data-bus could stay unnoticed if no precautions at a higher level (e.g., software) are taken.

4) Safety Standard Considerations: The SIL metrics define several levels for hardware safety integrity and fault tolerance. While SIL 2 only calls for the consideration of stuck-at faults, higher levels also require a diagnostic coverage fault model as

TABLE III
FAULT INJECTION INTO EMBEDDED PROCESSOR MEMORY RESULTS.

Benchmark	Inj. Faults	Calculations	Corrupted	Corrupted [%]
Basicmath_small (Client)	> 1460000	5637182	590432	10.47
Basicmath_large (Client)	> 791000	3048202	264451	8.68

TABLE IV
EVALUATION OF EMBEDDED PROCESSOR SELF-TEST PROCEDURES.

Self-test	Injection Type	Faults Injected	Detected [λ_S]	Fault Coverage [SFF]
Multiplier	Stuck-At 1	1000	1000	100%
Multiplier	Transient 1	1000	1000	100%
ALU	Stuck-At	1000	1000	100%
ALU	Transient	1000	1000	100%
Shifter	Stuck-At	1000	1000	100%
Shifter	Transient	1000	999	99.9%

well as the regard of soft-errors and dynamic cross influence of memory cells. To detect such problems during runtime, as required by safety standards, software self-tests have to be implemented. Without proper fault insertion testing it is not possible to check these self-tests for their effectivity. For the following test, we took an embedded processor core self-test collection from [29] aiming at testing the Multiplier, the ALU, and the Shifter unit of a processor. We injected faults into those units and obtained the resulting fault coverage values, which are necessary for safety certification, from the self-test results.

The results shown in Table IV mainly correspond to the coverage values given in [29] and distinguish between transient and stuck-at faults, which is required by the safety standard.

5) Experimental Conclusion: Fault resistance has become a system-wide issue in novel automotive systems that cannot be reduced to the evaluation of isolated modules anymore. Therefore, faults have been injected at ethernet protocol, controller and software level. The results show the importance of checksums and sequence numbers to provide communication and operation integrity. This gets especially evident when faults are injected into the ethernet controller's memory because protocol tests showed that this core is able to handle external faults, but not ones happening inside of the implementation. Finally, self-test routines for various parts of a system-on-chip have been tested for their fault detection efficiency.

VI. CONCLUSION

This work presented a systematic methodology for integrated power and fault emulation to efficiently evaluate the power consumption and fault resistance behavior of safety critical system-on-chip implementations. This combination of low-level hardware resource access with high-level emulation techniques allows for the direct evaluation of embedded digital control systems including its controlling software. Long-time testing experience from the smart-card industry sector has been used to develop a comprehensive hardware/software test strategy including a systems field bus connectivity. This methodology allows for the long-time multiple-bit fault investigations needed for comprehensive emulation-based testing as described in safety standards like IEC 61508.

The applicability of our approach has been evaluated using a common system-on-chip implementation including peripherals based on Ethernet bus connections. The retrieved experimental results proved its high performance and general adaptability to different system modules. Our future work includes the further investigation of structured RTL-level fault emulation for automated industrial system testing.

ACKNOWLEDGMENT

The authors would like to thank their project partners Infineon Technologies Austria AG and Austria Card GmbH.

REFERENCES

[1] H. Guzman-Miranda, L. Sterpone, M. Violante, M. Aguirre, and M. Gutierrez-Rizo, "Coping with the obsolescence of safety- or mission-critical embedded systems using FPGAs," *IEEE Trans. Ind. Electron.* , vol. 58, no. 3, pp. 814–821, 2011.
[2] E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, and M. Naouar, "FPGAs in industrial control applications," *IEEE Trans. Ind. Inf.*, vol. 7, no. 2, pp. 224–243, 2011.
[3] T. Enokido and M. Takizawa, "An integrated power consumption model for distributed systems," *IEEE Trans. Ind. Electron.*, vol. 60, no. 2, pp. 824–836, Feb. 2013.
[4] R. Leveugle, "A new approach for early dependability evaluation based on formal property checking and controlled mutations," in *Proc. IEEE 11th Int. Symp. on On-Line Testing*, 2005, pp. 260–265.
[5] H. Guzman-Miranda, M. Aguirre, and J. Tombs, "Noninvasive fault classification, robustness and recovery time measurement in microprocessor-type architectures subjected to radiation-induced errors," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 5, pp. 1514–1524, 2009.
[6] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: A new paradigm for power estimation," *DAC 2005*, pp. 700–705, 2005.
[7] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "A side channel attack countermeasure using system-on-chip power profile scrambling," in *Proc. IEEE 17th Int. On-Line Testing Symp.*, 2011, pp. 222–227.
[8] A. Krieg, C. Bachmann, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, "Accelerating early design phase differential power analysis using power emulation techniques," in *Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust*, 2011, pp. 81–86.
[9] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock, J. Haid, T. Aichinger, and C. Ulbricht, "Case study on multiple fault dependability and security evaluations," in *Microprocessors and Microsystems*. The Netherlands: Elsevier [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2012.05.016,
[10] E. Jenn, J. A. M. Rimén, J. Ohlsson, and J. Karlsson, "Fault injection into VHDL models: The MEFISTO tool," in *Proc. 24th Int. Symp. on Fault-Tolerant Computing*, 1994, pp. 66–75.
[11] J.-C. Baraza, J. Gracia, S. Blanc, D. Gil, and P.-J. Gil, "Enhancement of fault injection techniques based on the modification of VHDL code," *EEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 6, pp. 693–706, Jun. 2008.
[12] M. Valderas, M. Garcia, R. Cardenal, L. Ongil, and L. Entrena, "Advanced simulation and emulation techniques for fault injection," in *Proc. IEEE Ind. Electronics. Int. Symp.*, 2007, pp. 3339–3344.
[13] R. Leveugle, "Fault injection in VHDL descriptions and emulation," in *Proc. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Syst.*, 2002, pp. 414–419.
[14] L. Antoni, R. Leveugle, and B. Feher, "Using run-time reconfiguration for fault injection applications," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 5, pp. 1468–1473, Oct. 2003.
[15] H. Zheng, L. Fan, and S. Yue, "FITVS: A FPGA-BASED emulation tool for high-efficiency hardness evaluation," in *Proc. IEEE Int. Symp. on Parallel and Distributed Proc. with Applicat.*, 2008, pp. 525–531.
[16] J. Daveau, A. Blampey, G. Gasiot, J. Bulone, and P. Roche, "An industrial fault injection platform for soft-error dependability analysis and hardening of complex system-on-a-chip," in *Proc. IEEE Int. Rel. Physics Symp.*, 2009, pp. 212–220.
[17] R. Leveugle, "Early analysis of fault-based attack effects in secure circuits," *IEEE Trans. Comput.*, pp. 1431–1434, 2007.
[18] J. Abke, E. Böhl, and C. Henno, "Emulation based real time testing of automotive applications," in *Proc. 4th IEEE Int. On-Line Testing Workshop*, 1998, pp. 28–31.
[19] F. Corno, F. Esposito, M. S. Reorda, and S. Tosato, "Evaluating the effects of transient faults on vehicle dynamic performance in automotive systems," in *Proc. ITC Int. Test Conf.*, Oct. 2004, pp. 1332–1339.

[20] E. Armengaud, A. Steininger, and M. Horauer, "Towards a systematic test for embedded automotive communication systems," *IEEE Trans. Ind. Inf.*, vol. 4, no. 3, pp. 146–155, 2008.

[21] G. Grießnig, R. Mader, C. Steger, and R. Weiß, "Fault insertion testing of a novel CPLD-based fail-safe system," in *Proc. Conf. on Design, Automation and Test in Europe*, 2009, pp. 214–219.

[22] F. Baronti, E. Petri, S. Saponara, L. Fanucci, R. Roncella, R. Saletti, P. D'Abramo, and R. Serventi, "Design and verification of hardware building blocks for high-speed and fault-tolerant in-vehicle networks," *IEEE Trans. Ind. Electron.*, no. 99, pp. 1–1, 2011.

[23] A. Myaing and V. Dinavahi, "FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics," *IEEE Trans. Ind. Electron.*, vol. 58, no. 1, pp. 358–368, 2011.

[24] F. Restrepo-Calle, A. Martinez-Alvarez, F. Palomo, H. Guzman-Miranda, M. Aguirre, and S. Cuenca-Asensi, "Rapid prototyping of radiation-tolerant embedded systems on FPGA," in *Proc. Int. Conf. on Field Programmable Logic and Applicat.*, 2010, pp. 326–331.

[25] *Int. Standard Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-Related Systems*, IEC Std., IEC61508, 1998.

[26] *International Standard Road vehicles—Functional Safety, ISO Std*, ISO Std., ISO/DIS 26262, 2010.

[27] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE Int. Workshop on Workload Characterization*, 2001, pp. 3–14, WWC-4. 2001.

[28] E. Grochowski, D. Ayers, and V. Tiwari, "Microarchitectural dl/dt control," *IEEE Design Test Comput.*, vol. 20, no. 3, pp. 40–47, 2003.

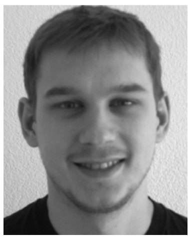
[29] A. Paschalis, D. Gizopoulos, N. Kranitis, M. Psarakis, and Y. Zorian, "Deterministic software-based self-testing of embedded processor cores," in *Proc. Conf. on Design, Automation and Test in Europe*, 2001, pp. 92–96.

Armin Krieg (S'10) received the Masters degree in telematics, from Graz University of Technology in 2008, focusing on microelectronics and system-on-chip design. Since 2010 he is working toward the Ph.D. degree in electrical engineering at the Institute for Technical Informatics, Graz University of Technology, in collaboration with Infineon Technologies Austria AG and Austria Card GmbH.

His research interests incorporate fault emulation as well as fault detection and recovery.

Christopher Preschern received the Masters degree in telematics from Graz University of Technology in 2011, focusing on software product lines, automation systems and IT-security. He is working toward the Ph.D. degree in electrical engineering at the at the Institute for Technical Informatics, Graz University of Technology.

His research focuses on safe and secure cyber-physical systems with special focus on automation systems.



Johannes Grinschgl (S'10) received the Masters degree in telematics from Graz University of Technology in 2008, focusing on microelectronics and telecommunications. Since 2010, he is working toward the Ph.D. degree in electrical engineering at the Institute for Technical Informatics, Graz University of Technology in collaboration with Infineon Technologies Austria AG and Austria Card GmbH.

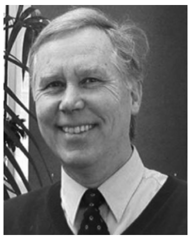
His research interests incorporate fault emulation as well as fault modelling.

Christian Steger (M'95) received the Dipl.-Ing. degree (equivalent to the American Master of Science) and the Dr. techn. degree in electrical engineering from Graz University of Technology, Austria, in 1990 and 1995, respectively.

He is Key Researcher at the Virtual Vehicle Competence Center (ViF, COMET K2), Graz, Austria. From 1989 to 1991, he was Software Trainer and Consultant at SPC Computer Training Ges.m.b.H., Vienna. Since 1992, he is Assistant Professor at the Institute for Technical Informatics, Graz University of Technology. He heads the HW/SW codesign group at the Institute for Technical Informatics. His research interests include embedded systems, HW/SW codesign, HW/SW coverfication, SOC, power awareness, smart cards, UHF RFID systems, multi-DSPs.

Christian Kreiner (M'99) graduated and received the Ph.D. degree in electrical engineering from Graz University of Technology, in 1991 and 1999, respectively.

From 1999 to 2007, he served as head of the R&D Department, Salomon Automation, Austria, focusing on software architecture, technologies, and processes for logistics software systems. He was in charge to establish a company-wide software product line development process and headed the product development team. There, he led and coordinated a long-term research programme together with the Institute for Technical Informatics of Graz University of Technology. There, he currently leads the Industrial Informatics and Model-based Architectures group. His research interests include systems and software engineering, software technology, and process improvement.



Reinhold Weiss (M'79) received the Dipl.-Ing. degree and the Dr.-Ing. degree (both in electrical engineering) and the Dr.-Ing.habil.degree (in realtime systems) from the Technical University of Munich, in 1968, 1972, and 1979, respectively.

Currently, he is a Professor of electrical engineering (Technical Informatics) and Head of the Institute for Technical Informatics, Graz University of Technology, Austria. His research interests focus on Embedded Distributed Real Time Architectures (distributed fault-tolerant systems, wearable and pervasive computing). He is a member of the International Editorial Board of the *Computers and Applications* (ISCA) journal.

Holger Bock received the Master's degree in electrical engineering from Graz University of Technology, Austria, in 1994.

From 1991 to 1998, he was working on concepts, software and hardware development, especially on VLSI-Design for cryptographic co-processors for smart cards (DES, ECC) at the Institute for Applied Information Processing and Communications Technologies (IAIK). In December 1998, he joined the team at Infineon's Development Center in Graz as a core competence for security. Since the beginning of 2001, he had been a member of the technology & innovations methodology team at Infineon's business group Chipcard & Security ICs, focusing on secure, especially DPA resistant, design methodologies for cryptographic hardware.

Josef Haid received the Master's degree in telematics and the Dr.-Ing. degree in electrical engineering from the Graz Technical University, Austria, in 2001 and 2003, respectively.

Presently, he is a Senior Staff Engineer at Infineon Technologies, Graz, Austria and is responsible for specification of low-power contactless smart cards. His interests include advanced digital design and low power design of hardware and software.

Acceleration of Fault Attack Emulation by Consideration of Fault Propagation

Armin Krieg, Johannes Grinschgl,
Christian Steger and Reinhold Weiss
Institute for Technical Informatics
Graz University of Technology, Austria
{armin.krieg, johannes.grinschgl,
steger, rweiss}@tugraz.at

Holger Bock, Josef Haid
Design Center Graz
Infineon Technologies Austria AG
{josef.haid, holger.bock}@infineon.com

***Abstract*—In recent years the number of deployed embedded systems increased significantly. These system-on-chips are widely used for high-availability as well as security applications. Therefore, the reliable operation of these devices plays a vital role and disturbed operation can lead to loss of confidence and trust. To ensure correct operation during random or intentional fault events, injection techniques for system simulation and emulation have been presented. The targeted use of these approaches is often difficult because of the device complexity and the lack of knowledge about internal processes after a fault has been activated. To improve the current state-of-the-art in this field this paper presents fault propagation analysis and hardware checker generation techniques based on static VHDL code analysis. These help to gain a deeper understanding of system internal propagation paths and their influence on normal operation. Physical layout data is included to enable the mapping of a fault attack location to its corresponding logic gates. Hardware checkers enable higher fault injection evaluation efficiency by removing masked system parts from the target space.**

I. Introduction

The continuing introduction of smart-cards into the monetary cycle and for personal ID applications created a viable target for criminal subjects [1]. This led to intense research concerning attack methods and corresponding countermeasures to prevent the leakage of vital information to the environment. Basically, two different kinds of attacks can be differentiated, passive ones, listening to the device's behavior and active ones, influencing normal operation by the introduction of faults. Such fault-based attacks are used to drive the system into an unintended state to disable existing countermeasures and to disrupt the normal mode of cryptographic operation [2].

These issues resulted in a wide range of research concerning fault simulation and emulation techniques. Using these methodologies software and hardware developers should be enabled to test their implementations for its robustness against operational errors. These evaluation platforms have been used to replicate the effects of soft and hard errors on embedded system implementations. Unfortunately there is a distinctive lack of research concerning the link between low-level faults and high-level erroneous results. Especially when testing large system-on-chips the success of a fault injection can only be seen through errors manifesting themselves on system outputs or memories. Faults that have been temporarily masked could lead to errors during later operation and hence, could result in false evaluation outcomes.

Therefore, there is a strong need for static code analysis techniques to determine fault propagation paths and to provide efficient injection platforms. In this work a novel methodology is presented, combining knowledge about fault attack locations and internal propagation paths to enable real-time emulation of logic fault effects on system operation (depicted in Figure 1). The incorporation of the precise fault attack location is specifically important in case of accurate laser attacks as used by test laboratories and adversaries [3].

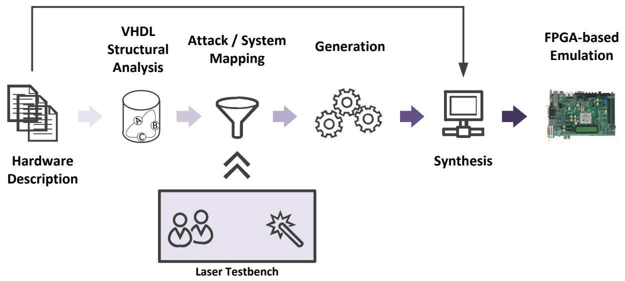


Fig. 1. Global fault attack emulation flow

The main contributions of this work are:

- A static VHDL analysis flow to provide the automatic generation of fault injection emulation support modules.
- Introduction of a novel fault attack propagation analysis flow for post-laser injection evaluation analysis.

This paper is structured as follows. First, Section II will give a brief introduction into static analysis methods of VHDL code. Followed by Section III introducing our methodology for efficient emulation of fault-based attacks. Section IV compares our methodology and results with recent developments in this area. In Section V experimental results are shown to prove the applicability of our approach. Finally, Section VI concludes this work and gives a short overview over our future work.

978-1-4673-2845-6/12/$31.00 © 2012 IEEE

II. Static system analysis

For security evaluations not only information of the basic system structure is sufficient, therefore our approach includes also information from the device layout and the laser attack coordinates used on a physical test bench.

A. Static code analysis

The first step includes the automatized extraction of the signal and variable dependency graphs. This process consists of three stages.

VHDL File Parsing : Existing digital design flows already include simple interdependency checking mechanisms to optimize their compilation processes. While our proposed VHDL parsing engine is able to process large VHDL file compilation, such pre-processing stages can be used to reduce the parsing effort needed during this stage. Basically all significant VHDL file elements are extracted e.g. signals, variables, process and constants. This includes eventual assignment targets and sources or elements inside sensitivity-lists.

Local Dependency Extraction : After all significant elements are available inside an internal database, local dependency extraction can be started. This process consists of processing all assignment source and target lists to generate local (module-internal) dependency graphs.

Global Dependency Extraction : The result of the local extraction step are individual objects containing dependency graphs. These are finally connected to gain a complete system graph including all found objects and their relations.

B. Layout analysis

After the hardware description has been analyzed a connection between layout position data and the laser attack coordinates has to be established. Again this flow includes three basic stages that have to be executed to provide an accurate fault model.

Layout-Location-Data : The first step contains the extraction of the logic gate positions inside the finalized layout. This information is included in the DEF-file produced by layout tools such as those provided by the Cadence digital implementation flow.

Laser-Grid-Positions : These laser attack positions are provided by the laser bench test engineer team. Basically every order of positions can be used, most likely this will be a systematic grid.

Attack-Position-Mapping : A mapping process is finally used to identify logic gates that are affected by the chosen laser attack. The effect radius depends on the accuracy of the chosen laser beam. Additionally knowledge of physical design engineers can be included into the determination of affected gates. Such information is depending on laser spot size, shot duration and pulse strength.

C. Location to RTL mapping

The results of the former two process steps now has to be connected to retrieve a connection between the physical layout and the higher-level hardware description. This mapping process consists of three stages.

Signal Filtering : The layout processing stage resulted in a list of affected gates and signals and their corresponding positions on the layout. This information is now filtered, because not every element known at the RTL level is also visible after synthesis.

Signal-to-Node Mapping : Now only elements are available that also visible in the RTL description, therefore it is possible to map the physical positions to internal nodes existing in the dependency graphs.

Node-to-Graph Mapping : Finally, these nodes are mapped into the global design dependency graph to enable further dependency analysis.

After RTL and layout extraction and mapping processes, a complete database is available containing affected elements and their relations. This information can now be used to generate checker modules and the optimum positions for saboteur blocks to emulate laser attack effects.

III. Fault attack emulation

An accurate physical fault model now has been established and can be included into an FPGA-based emulation platform. The flow starting with the generation of saboteur modules and resulting in evaluation conclusions is divided into four steps.

Generation Process : First, saboteurs and checker modules have to be generated depending on the amount of signals that have to be disturbed or monitored. This information is taken from the local dependency graphs of the selected attack targets. Manipulations are done using free configurable saboteurs and system properties are monitored using trigger modules (similar to the approach taken in [4]).

Placement Process : During the placement process the generated modules are placed into the RTL description of the target system and are connected to analysis support blocks. Such blocks include the fault injection controller which is handling all the saboteurs and controlling the injection process.

Injection Process : At this stage the RTL source code has been prepared and synthesized to fit into the FPGA of the evaluation platform. The fault injection control is configured with all selected attack patterns and starts with the evaluation the system-under-test. Results from these injection campaigns can be directly derived from placed checker modules or by monitoring selected outputs and memories.

Analysis Process : All stored evaluation results are finally stored onto a connected PC and analyzed to identify unprotected information paths. Also this information helps to understand possible failures identified during laser bench tests that are part of system certification.

The final emulation architecture includes a controlling micro-processor to ensure a fast and independent injection process as shown in Figure 2. This system-external controller can also be used to efficiently monitor certain aspects of the target system like state signals or memory buses. Fault attack effects are mapped to the design using saboteur modules that emulate various signal manipulation types.

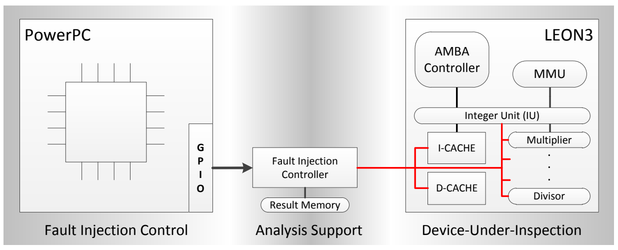


Fig. 2. FPGA-based fault injection emulation architecture - High performance fault injection evaluation for secure software verification

IV. Related Work

A. Fault simulation and emulation

An early implementation of such a simulation environment under fault conditions was called MEFISTO [5]. In case of MEFISTO the behavior of the system is directly influenced by small saboteur modules placed between target signal and variable sources and sinks. The following years various improvements to this simulation approach have been presented, mainly to improve simulation speed, e.g., the VFIT tool presented in [6]. In the last years also a wide range of SystemC-based approaches have been proposed [7], [8]. At the cost of injection flexibility hardware-accelerated FPGA-based approaches have been presented to increase performance and to enable large-scale dependability investigations [9]. In [10], authors showed that this single-fault model is not sufficient if intentional faults, as expected in security evaluations, are concerned. Specifically targeting fault simulation of optical fault attacks, an approach has been presented by the authors of [11] and [12]. As test costs demanded by certification labs increased dramatically, high speed software verification using hardware-accelerated methods became important. A modular methodology for such an emulation-based platform has been shown in [4] and hardware checker generation has been presented in [13].

B. Fault propagation modeling

For many years low-level logic evaluation for soft-error reliability (SER) has been done using slow but very accurate SPICE-based simulations. Few years ago the authors of [14] introduced various new techniques based on binary decision diagrams to improve such investigations performance-wise. Similarly, a signature-based approach has been introduced in [15] to analyze SER and logic masking in combinational and sequential circuits. Especially of interest in the security domain is the work proposed in [16] specifically targeting multiple transient faults as they would be expected during complex fault-attacks.

C. Static VHDL code analysis

Timing and particularly worst-case execution time (WCET) analysis is an important part of the evaluation of hard real-time and safety systems. To automate this error-prone process, which includes a hard engineering effort, the authors of [17] presented a frame work for the static analysis of VHDL code. In this early work the VHDL description is treated like a sequential program to derive a control flow description in CRL2. The same research group introduced an abstraction-aware compiler for such hardware description models in [18]. Finally, this semi-automatic process of hardware model timing analyses has been completed with the framework shown in [19]. In the security domain not only control flow, but also flow of information is of essential importance. Therefore, a state-machine-based approach has been presented in [20] to determine information flow at a higher abstraction level.

V. Experimental Results

Our proposed methodology is implemented and demonstrated using an open source implementation of the SPARC v8 architecture developed by Aeroflex Gaisler. This system-on-chip example has been synthesized using Xilinx ISE software and is functionally tested on the ML507 evaluation board.

A. Attack mapping

The first step includes the mapping of a laser-based attack to the hardware description of the emulation system. To increase demonstration quality the chosen attack-path is very short and locally limited. Such a sample file includes information about laser impact radius and targeted coordinates.

This file is parsed and influenced hardware elements are identified using the DEF-file of the place&route process. In this case a 32-bit wide register inside the AHB-controller of the LEON3 processor has been disturbed. The resulting dependency graph for this node is depicted in Figure 3. Full black arrows mean signal or variable assignments from one node to another. Dotted lines indicate that the target node is sensitive on the given source node because the assignment happens inside a process. Bold grey lines depict enable relation-ships if the assignment is located inside an if-structure.

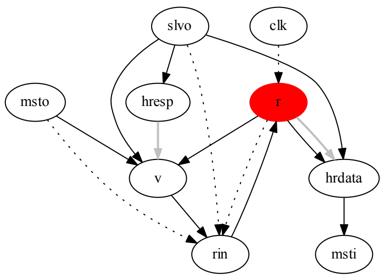


Fig. 3. Simplified isolated dependency graph of influenced node

The influenced node is part of a ring-dependency, meaning that the reading of **rin** is triggered by the two inputs **msto** and **slvo** while writing into node **r** is triggered only by the clock-signal. The backward connection is realized via variable **v**, which is the only externally written part. The writing of **v** is enabled via variable **hresp**. Finally the result residing in **r** is written to the output **msti**. This information can now be used for the generation of hardware modules for efficient emulation of this chosen attack. This includes saboteur modules to manipulate this register in a way our chosen laser impulse would and hardware checkers to enable these saboteurs only when this attack would not have been masked.

B. Attack emulation and effectiveness evaluation

Such isolated dependency graphs are now processed for the further generation of VHDL hardware blocks like saboteurs or checkers. In the former case the simple manipulation of the targeted element can be relatively simply implemented and several approaches using saboteurs have been shown in Section IV. On the other hand, checker modules allow more in-depth analysis of internal processes after fault activation.

In this combinatorial block a change of **r** directly triggers the writing of **rin**. The graph in this case (record types) is a bit misleading because only a very small part of **r** is influenced. One control signal of this record is also enabling the writing of **hrdata** and hence the fault propagates to the output. Such a checker generated trace can be seen in Figure 4, visualizing that there are large parts of the execution that are insensitive.

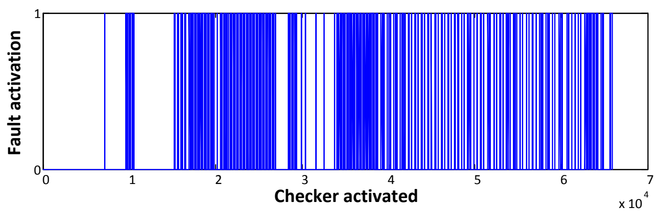


Fig. 4. Checker result trace - CPU startup and AES encryption

As seen in Table I only 11 percent of all accesses to this register would lead to a propagation of this fault because in all other cases there is no enable.

TABLE I
CHECKER EVALUATION RESULT FOR A SINGLE ATTACKED NODE

Checker Name	Checker activ.	Fault sens.	Fault sens. [%]
AHBCTRL_HRDATAS	65896	7261	11

Besides the observation of node-activity of our attacked system blocks, we are enabled to implement saboteur modules at this point to emulate signal manipulations. In such a case the results of the placed support modules, like these checker blocks, will provide input to the injection control to trigger attacks at fault sensitive points of time.

VI. Conclusion

In this work a novel methodology for the efficient generation of hardware checker modules and fault emulation support blocks is presented. Static VHDL code analysis methods have been applied to extract fault propagation paths from the investigated system and combined with information gained after physical evaluation. Physical properties of the laser point of attack are abstracted into a single region of logic interaction. This way a gap between physical attacks using laser impulses and the RTL design is closed to enable a better understanding of effects caused by physical fault injection.

Our approach has been tested using open available source of a system-on-chip implementation and its modular software design allows for the integration into existing design tools.

Acknowledgments

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation, and Technology, which funded the Power-Modes Project under the FIT-IT contract FFG 825749. We would also like to thank our project partners Infineon Technologies Austria AG and Austria Card GmbH.

References

[1] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, “Security in embedded systems: Design challenges,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 461–491, 2004.
[2] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Moderator-Ravi, “Security as a new dimension in embedded system design,” in *Design Automation Conference*. ACM, 2004, pp. 753–760.
[3] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, “The sorcerer's apprentice guide to fault attacks,” *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006.
[4] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock, and J. Haid, “Modular fault injector for multiple fault dependability and security evaluations,” in *Euromicro DSD*. IEEE, 2011, pp. 550–557.
[5] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson, “Fault injection into VHDL models: the MEFISTO tool,” in *Fault-Tolerant Computing, Twenty-Fourth International Symposium on*. IEEE, 1994, pp. 66–75.
[6] J. Baraza, J. Gracia, S. Blanc, D. Gil, and P. Gil, “Enhancement of fault injection techniques based on the modification of VHDL code,” *VLSI, IEEE Trans. on*, vol. 16, no. 6, pp. 693–706, 2008.
[7] K. Rothbart, U. Neffe, C. Steger, R. Weiss, E. Rieger, and A. Muehlberger, “High level fault injection for attack simulation in smart cards,” in *Test Symposium, 2004. 13th Asian*. IEEE, 2004, pp. 118–121.
[8] S. Misera, H. Vierhaus, and A. Sieber, “Simulated fault injections and their acceleration in SystemC,” *Microprocessors and Microsystems*, vol. 32, no. 5-6, pp. 270–278, 2008.
[9] D. de Andrés, J. Ruiz, D. Gil, and P. Gil, “Fault emulation for dependability evaluation of VLSI systems,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 4, pp. 422–431, 2008.
[10] R. Leveugle, “Early analysis of fault-based attack effects in secure circuits,” *Comp., IEEE Trans. on*, vol. 56, no. 10, pp. 1431–1434, 2007.
[11] S. Skorobogatov, “Semi-invasive attacks-a new approach to hardware security analysis,” *Technical report*, 2005.
[12] H. Li and S. Moore, “Security evaluation at design time against optical fault injection attacks,” *Information Security, IEE Proceedings*, vol. 153, no. 1, pp. 3 – 11, march 2006.
[13] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, A. Genser, H. Bock, and J. Haid, “Characterization and handling of low-cost micro-architectural signatures in MPSoCs,” in *ETS, IEEE*, 2012, pp. 1 –6.
[14] N. Miskov-Zivanov and D. Marculescu, “Modeling and optimization for soft-error reliability of sequential circuits,” *CAD of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 5, pp. 803–816, 2008.
[15] S. Krishnaswamy, S. Plaza, I. Markov, and J. Hayes, “Signature-based SER analysis and design of logic circuits,” *CAD of Integrated Circuits and Systems, IEEE Trans. on*, vol. 28, no. 1, pp. 74–86, 2009.
[16] N. Miskov-Zivanov and D. Marculescu, “Multiple transient faults in combinational and sequential circuits: a systematic approach,” *CAD of ICs and Systems, IEEE Trans. on*, vol. 29, no. 10, pp. 1614–1627, 2010.
[17] M. Schlickling and M. Pister, “A Framework for Static Analysis of VHDL Code,” in *7th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis*, C. Rochange, Ed., Dagstuhl, Germany, 2007.
[18] M. Maksoud, M. Pister, and M. Schlickling, “An abstraction-aware compiler for VHDL models,” in *Computer Engineering & Systems. International Conference on*. IEEE, 2009, pp. 3–9.
[19] M. Schlickling and M. Pister, “Semi-automatic derivation of timing models for WCET analysis,” in *ACM SIGPLAN Notices*, vol. 45, no. 4. ACM, 2010, pp. 67–76.
[20] X. Li, M. Tiwari, B. Hardekopf, T. Sherwood, and F. Chong, “Secure information flow analysis for hardware design: Using the right abstraction for the job,” in *Program. Lang. and Anal. for Sec.* ACM, 2010.

Bibliography

[1] J. Browne, “Dawn Of The M2M Age.” online, 2011. http://chipdesignmag.com/sld/ferro/2011/04/28/dawn-of-the-m2m-age/.

[2] ITRS, “International Roadmap for Semiconductors (2011 Edition).” online. http://www.itrs.net/Links/2011ITRS/Home2011.htm.

[3] H. Foster, “Challenges of Design and Verification in the SoC Era.” online, 2011. http://blogs.mentor.com/verificationhorizons/blog/author/hfoster/.

[4] J. Grinschgl and A. Krieg, “POWER-MODES - Power Emulator and Model based Dependability and Security evaluation platform,” tech. rep., Institute for Technical Informatics, Graz University of Technology, 2012.

[5] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, H. Bock, and J. Haid, “POWER-MODES - POWer EmulatoR and MOdel based Dependability and Security evaluations,” *Reconfigurable Technology and Systems (TRETS), ACM Transactions on*, vol. accepted for publication, 2012.

[6] A. Krieg, C. Bachmann, J. Grinschgl, C. Steger, R. Weiss, and J. Haid, “Accelerating early design phase differential power analysis using power emulation techniques,” in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*, pp. 81–86, IEEE, 2011.

[7] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, H. Bock, and J. Haid, “System side-channel leakage emulation for HW/SW security coverification of MPSoCs,” in *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2012 IEEE 15th International Symposium on*, pp. 139–144, IEEE, 2012.

[8] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, H. Bock, and J. Haid, “Acceleration of Fault Attack Emulation by Consideration of Fault Propagation,” in *Field-Programmable Technology (FPT), 11th International Conference on*, IEEE, 2012.

[9] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, A. Genser, H. Bock, and J. Haid, “Characterization and handling of low-cost micro-architectural signatures in MPSoCs,” in *Test Symposium (ETS), 2012 17th IEEE European*, pp. 1 –6, 2012.

[10] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, H. Bock, and J. Haid, “PROCOMON - An Automatically Generated Predictive Control Signal Monitor,” in *Digital System Design (DSD), 15th Euromicro Conference on*, pp. 654–660, IEEE, 2012.

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

[12] Aeroflex Gaisler, "Leon3 Processor." online, September 2012. http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53.

[13] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock, J. Haid, T. Aichinger, and C. Ulbricht, "Case Study on Multiple Fault Dependability and Security Evaluations," *Microprocessors and Microsystems*, 2012.

[14] A. Krieg, J. Grinschgl, C. Steger, R. Weiss, H. Bock, and J. Haid, "Power And Fault Emulation For Software Verification and System Stability Testing in Safety Critical Environments," *IEEE Transactions on Industrial Informat*, vol. under review, 2012.

[15] Z. Alkhalifa, V. Nair, N. Krishnamurthy, and J. Abraham, "Design and evaluation of system-level checks for on-line control flow error detection," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 10, no. 6, pp. 627–641, 1999.

[16] N. Oh, P. Shirvani, and E. McCluskey, "Control-flow checking by software signatures," *Reliability, IEEE Transactions on*, vol. 51, no. 1, pp. 111–122, 2002.

[17] J. Smolens, B. Gold, J. Kim, B. Falsafi, J. Hoe, and A. Nowatzyk, "Fingerprinting: bounding soft-error detection latency and bandwidth," in *ACM SIGARCH Computer Architecture News*, vol. 32, pp. 224–234, ACM, 2004.

[18] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[19] T. Ye, L. Benini, and G. De Micheli, "Packetized on-chip interconnect communication analysis for MPSoC," in *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pp. 344–349, IEEE, 2003.

[20] EUROSMART, "Eurosmart expects over 7 billion smart secure devices to be shipped in 2012," April 2012. http://www.eurosmart.com/index.php/publications/market-overview.html.

[21] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low power methodology manual: for system-on-chip design.* Springer Publishing Company, Incorporated, 2007.

[22] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power CMOS digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 4, pp. 473–484, 1992.

[23] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. Kim, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, no. 2, pp. 43–52, 2005.

[24] H. Guzman-Miranda, M. Aguirre, and J. Tombs, "Noninvasive fault classification, robustness and recovery time measurement in microprocessor-type architectures subjected to radiation-induced errors," *Instrumentation and Measurement, IEEE Trans. on*, vol. 58, no. 5, pp. 1514–1524, 2009.

[25] R. Leveugle, "Fault injection in VHDL descriptions and emulation," in *Defect and Fault Tolerance in VLSI Systems, 2000. Proceedings. IEEE International Symposium on*, pp. 414–419, IEEE, 2002.

[26] A. Genser, C. Bachmann, J. Haid, C. Steger, and R. Weiss, "An emulation-based real-time power profiling unit for embedded software," in *Systems, Architectures, Modeling, and Simulation, 2009. SAMOS'09. International Symposium on*, pp. 67–73, IEEE, 2009.

[27] V. Kozhikkottu, R. Venkatesan, A. Raghunathan, and S. Dey, "VESPA: Variability emulation for System-on-Chip performance analysis," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pp. 1–6, IEEE, 2011.

[28] S. Reda and A. Nowroz, "Power Modeling and Characterization of Computing Devices: A Survey," *Foundations and Trends® in Electronic Design Automation*, vol. 6, no. 2, pp. 121–216, 2012.

[29] J. Coburn, S. Ravi, and A. Raghunathan, "Power emulation: a new paradigm for power estimation," in *Proceedings of the 42nd annual Design Automation Conference*, pp. 700–705, ACM, 2005.

[30] M. Ghodrat, K. Lahiri, and A. Raghunathan, "Accelerating system-on-chip power analysis using hybrid power estimation," in *Design Automation Conference, 2007. DAC'07. 44th ACM/IEEE*, pp. 883–886, IEEE, 2007.

[31] A. Bhattacharjee, G. Contreras, and M. Martonosi, "Full-system chip multiprocessor power evaluations using FPGA-based emulation," in *Low Power Electronics and Design (ISLPED), 2008 ACM/IEEE International Symposium on*, pp. 335–340, IEEE, 2008.

[32] C. Bachmann, A. Genser, C. Steger, R. Weiss, and J. Haid, "Automated power characterization for run-time power emulation of soc designs," in *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pp. 587–594, IEEE, 2010.

[33] C. Bachmann, A. Genser, C. Steger, R. Weiß, and J. Haid, "Accelerating embedded software power profiling using run-time power emulation," *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, pp. 186–195, 2010.

[34] A. Genser, C. Bachmann, C. Steger, R. Weiss, and J. Haid, "Power emulation based DVFS efficiency investigations for embedded systems," in *System on Chip (SoC), 2010 International Symposium on*, pp. 173–178, IEEE, 2010.

[35] D. Shumov and P. L. Montgomery, "Side Channel Leakage Profiling in Software," in *COSADE 2010*, February 2010.

[36] J. den Hartog and E. De Vink, "Virtual Analysis and Reduction of Side-Channel vulnerabilities of smartcards," *Formal Aspects in Security and Trust*, pp. 85–98, 2005.

[37] C. Thuillet, P. Andouard, and O. Ly, “A smart card power analysis simulator,” in *Computational Science and Engineering, 2009. CSE’09. International Conference on*, vol. 2, pp. 847–852, Ieee, 2009.

[38] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, “McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures,” in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 469 –480, 2009.

[39] F. Regazzoni, A. Cevrero, F. Standaert, S. Badel, T. Kluter, P. Brisk, Y. Leblebici, and P. Ienne, “A design flow and evaluation framework for DPA-resistant instruction set extensions,” *Cryptographic Hardware and Embedded Systems-CHES 2009*, pp. 205–219, 2009.

[40] M. Bucci, R. Luzzi, F. Menichelli, R. Menicocci, M. Olivieri, and A. Trifiletti, “Testing power-analysis attack susceptibility in register-transfer level designs,” *IET 2007*, vol. 1, no. 3, pp. 128–133, 2007.

[41] R. Leveugle and K. Hadjiat, “Multi-level fault injections in VHDL descriptions: alternative approaches and experiments,” *Journal of Electronic Testing*, vol. 19, no. 5, pp. 559–575, 2003.

[42] K. Rothbart, U. Neffe, C. Steger, R. Weiss, E. Rieger, and A. Muehlberger, “High level fault injection for attack simulation in smart cards,” in *Test Symposium, 2004. 13th Asian*, pp. 118–121, IEEE, 2004.

[43] J.-C. Baraza, J. Gracia, S. Blanc, D. Gil, and P.-J. Gil, “Enhancement of Fault Injection Techniques Based on the Modification of VHDL Code,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, pp. 693 –706, june 2008.

[44] M. Valderas, M. Garcia, R. Cardenal, L. Ongil, and L. Entrena, “Advanced Simulation and Emulation Techniques for Fault Injection,” in *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pp. 3339–3344, IEEE, 2007.

[45] M. Sauer, V. Tomashevich, J. Muller, M. Lewis, A. Spilla, I. Polian, B. Becker, and W. Burgard, “An FPGA-based framework for run-time injection and analysis of soft errors in microprocessors,” in *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*, pp. 182–185, IEEE, 2011.

[46] O. Goloubeva, M. Rebaudengo, M. Sonza Reorda, and M. Violante, “Soft-error detection using control flow assertions,” in *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*, pp. 581–588, IEEE, 2003.

[47] R. Venkatasubramanian, J. Hayes, and B. Murray, “Low-cost on-line fault detection using control flow assertions,” in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pp. 137–143, IEEE, 2003.

[48] V. Iyengar and L. Kinney, “Concurrent fault detection in microprogrammed control units,” *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 810–821, 1985.

[49] S. Mao and T. Wolf, “Hardware support for secure processing in embedded systems,” *Computers, IEEE Transactions on*, vol. 59, no. 6, pp. 847–854, 2010.

[50] S. Lukovic, P. Pezzino, and L. Fiorin, “Stack Protection Unit as a step towards securing MPSoCs,” in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pp. 1–4, IEEE, 2010.

[51] K. Patel, S. Parameswaran, and R. Ragel, “Architectural Frameworks for Security and Reliability of MPSoCs,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, no. 99, pp. 1–14, 2010.

[52] J. Smolens, B. Gold, J. Hoe, B. Falsafi, and K. Mai, “Detecting emerging wearout faults,” in *Proc. of Workshop on SELSE*, 2007.

[53] O. Khan and S. Kundu, “Hardware/Software Codesign Architecture for Online Testing in Chip Multiprocessors,” *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, no. 5, pp. 714 –727, 2011.

[54] M. Bozzano and A. Villafiorita, “The FSAP/NuSMV-SA safety analysis platform,” *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 9, no. 1, pp. 5–24, 2007.

[55] J. Ezekiel and A. Lomuscio, “Combining fault injection and model checking to verify fault tolerance in multi-agent systems,” in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 113–120, 2009.

[56] S. Morimoto, S. Shigematsu, Y. Goto, and J. Cheng, “Formal verification of security specifications with common criteria,” in *Proceedings of the 2007 ACM symposium on Applied computing*, pp. 1506–1512, ACM, 2007.

[57] B. Chetali and Q. Nguyen, “Industrial use of formal methods for a high-level security evaluation,” *FM 2008: Formal Methods*, pp. 198–213, 2008.

[58] M. Boulé and Z. Zilic, *Generating hardware assertion checkers: for hardware verification, emulation, post-fabrication debugging and on-line monitoring.* Springer Verlag, 2008.

[59] Z. Andraus and K. Sakallah, “Automatic abstraction and verification of Verilog models,” in *Proceedings of the 41st annual Design Automation Conference*, pp. 218–223, ACM, 2004.

[60] A. Karputkin, R. Ubar, M. Tombak, and J. Raik, “Automated correction of design errors by edge redirection on High-Level Decision Diagrams,” in *Quality Electronic Design (ISQED), 2012 13th International Symposium on*, pp. 686–693, IEEE, 2012.

[61] M. Schlickling and M. Pister, “A Framework for Static Analysis of VHDL Code,” in *7th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis* (C. Rochange, ed.), (Dagstuhl, Germany), 2007.

[62] M. Maksoud, M. Pister, and M. Schlickling, "An abstraction-aware compiler for VHDL models," in *Computer Engineering & Systems, 2009. ICCES 2009. International Conference on*, pp. 3–9, IEEE, 2009.

[63] M. Schlickling and M. Pister, "Semi-automatic derivation of timing models for WCET analysis," in *ACM SIGPLAN Notices*, vol. 45, pp. 67–76, ACM, 2010.

[64] X. Li, M. Tiwari, B. Hardekopf, T. Sherwood, and F. Chong, "Secure information flow analysis for hardware design: Using the right abstraction for the job," in *5th ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*, p. 8, ACM, 2010.

[65] X. Li, M. Tiwari, J. Oberg, V. Kashyap, F. Chong, T. Sherwood, and B. Hardekopf, "Caisson: a hardware description language for secure information flow," *Proceedings of Programming Language Design and Implementation (PLDI 2011)*, 2011.

[66] N. Miskov-Zivanov and D. Marculescu, "Modeling and optimization for soft-error reliability of sequential circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 5, pp. 803–816, 2008.

[67] S. Krishnaswamy, S. Plaza, I. Markov, and J. Hayes, "Signature-based SER analysis and design of logic circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 1, pp. 74–86, 2009.

[68] N. Miskov-Zivanov and D. Marculescu, "Multiple transient faults in combinational and sequential circuits: a systematic approach," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 10, pp. 1614–1627, 2010.

[69] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards.* Springer Verlag, 2007.

[70] J. Takahashi, T. Fukunaga, and K. Yamakoshi, "DFA Mechanism on the AES Key Schedule," in *Fault Diagnosis and Tolerance in Cryptography (FDTC). Workshop on*, pp. 62 –74, sept. 2007.

[71] D. Arora, S. Ravi, A. Raghunathan, and N. Jha, "Secure embedded processing through hardware-assisted run-time monitoring," in *Design, Automation and Test in Europe, Proceedings of the conference on*, pp. 178–183, IEEE Computer Society, 2005.

[72] Institute for Technical Informatics, Graz University of Technology, "META[:SEC:] - Mobile Energy-efficient Trustworthy Authentication Systems with Elliptic Curve based SECurity." FFG FIT-IT Cooperative Research Project 829586, 2011.