Académie de Montpellier

Université Montpellier II

- Sciences et Techniques du Languedoc -

Thèse

pour obtenir le grade de

Docteur de l'université Montpellier II

Discipline :	*Génie Informatique, Automatique et Traitement du signal*
Formation doctorale :	*Systèmes Automatiques et Microélectroniques*
Ecole doctorale :	*Information, Structures, Systèmes*

Securing embedded systems based on FPGA technologies

Présentée et soutenue publiquement par

Florian Devic

le 6 juillet 2012

Jury :

Rapporteurs :	Lilian Bossuet	-	Maitre de conférences, Laboratoire Hubert Curien
	Guy Gogniat	-	Professeur, Lab-STICC
Président :	Christophe Jégo	-	Professeur, IMS
Examinateurs :	Pascal Benoit	-	Maitre de conférences, LIRMM
	Olivier Détour	-	Société Netheos
Directeur de thèse :	Lionel Torres	-	Professeur, LIRMM

L'art de la citation est l'art de ceux qui ne savent pas réfléchir par eux-même.

François-Marie Arouet (Voltaire) 1694 - 1778

à Camille,

Acknowledgments

First of all, I would like to thank the most beautiful discovery that I have made: Camille.

Then, my thanks go to my parents and all my family, including Camille's family and more particularly to Annie for its paper reviews.

I am very grateful to my jury members for their valuable effort and time to be my PhD examination committee: Pascal Benoit, Lilian Bossuet, Olivier Détour, Guy Gogniat, Christophe Jégo and Lionel Torres. I also want to thank my thesis supervisor, Lionel Torres, and my supervisor in Netheos, Benoît Badrignans, for the three years they devoted to my work and theirs constructive comments which enabled me to value my work effectively.

My thanks to the SecReSoC project team, Benoît Badrignans, Pascal Benoit, Pascal Cotret, Jean-Luc Danger, Viktor Fischer, Robert Fouquet, Lubos Gaspar, Guy Gogniat, Houssem Maghrebi and Lionel Torres. I would like to thank also the ANR -French National Research Agency- for its support in the frame of the SecReSoC project (ARPEGE 2009 program, ANR-09-SEGI-013).

I also thank the Netheos team with whom I worked and shared good times: Nicolas Badenne, Benoît Badrignans, Fabien Barbero, Jean-Christophe Bernard, Tobias Bircher, Adrien Bresson, Jim Buzon, Lionel Chanson, Franck Desfours, Olivier Détour, David Emo, Xavier Facélina, Jeremy Ferrandi, Christine Igounet, Paul Merlin, Laurent Morel, Wilfrid Oddou, Cyril Pascal, Emmanuel Peretto, Michel Prunet, Augustin Sarr, Christophe Souvignier, Jean-Michel Tonneau and Frédéric Yhuel.

I must also thank all the PhD students, post-docs and research engineers of microelectronics department that I attended during my thesis: Lyonel Barthe, Morgan Bourrée, Kaouthar Bousselam, Florent Bruguier, Rémi Busseuil, Luis Vitorio Cargnini, Samuel Cohet, Jérémie Crenne, Souha Hacine, Carolina Momo Metzler, Francois Poucheret, Jean Da Rolt, Olivier Rossel, Jérémie Salles, Zhenzhou Sun, Duc Anh Tran, Georgios Tsiligiannis, Miroslav Valka, Bruno Vaquie and all others that I forget...

I sincerely thank my classmates at Polytech'Montpellier and above all, awesome friends: Jérôme Barbaras, Amine Brizini, Alain Delvare, Victor Gasia, Anthony Lihard, Danny Lihard, Julien Lioure, Julien Ponnou.

Special thanks to Jojo and Panpan to have maintained my garden during my travels in conferences. Without forgetting Fifi who has since died.

I would to thanks Alice, Bob, Carol and Dave for all the time they spent to explain to me cryptography from the basis to the most complicated protocol. I would to thanks also Eve, Mallory, Oscar and Trudy who have made my task difficult

but without whom I would never have found the courage and the desire to work in security domain.Thank you also to Isaac, Ivan, Justin, Matilda, Nestor, Peggy, Plod, Trent, Vanna, Victor and Walter.

And finally, I thank Olivier Commowick for his latex template and Dr Nicola L. C. Talbot (Mrs Nicola Cawley) for her glossaries package and her reactivity and help.

Contents

List of Figures

List of Tables

Introduction

Context

Nowadays embedded systems are an integral part of our lives. Not a day goes by without using embedded systems and all major companies now rely on mobility and telecommunications. They can be, for example, smartphones, phones, computers, tablets, home appliances, GPSs (Global Positioning Systems), music players, car electronics, set-top boxes, video equipments, game consoles, medical facilities, ... They allow us to easily access services everyday: geolocation, communication, online banking, web browsing, gaming, remote working, ... All these devices have access to both personal and professional, who can be sensitive: banking informations, emails, photos, passwords, location data, ...

To realize embedded systems, designers have two options. They can design specific circuits for the developed application: ASICs (Application-Specific Integrated Circuits), or use reconfigurable circuits: FPGAs (Field-Programmable Gate Arrays). The latter correspond to circuits whose architecture is configurable depending on the application to be developed. The use of these in embedded systems is the result of a process of evolution and innovation for several decades.

In the 1980s, the U.S. company Xilinx represented by the two Co-Founders, Ross Freeman and Bernard Vonderschmitt, and installed in California in Silicon Valley was a forerunner in the field by launching the first FPGA business, the XC2064. This FPGA is composed by 64 CLBs (Configurable Logic Blocks) containing two 3-input LUTs (Look Up Tables). Xilinx will be followed in early 1990 by its competitors Altera and Actel.

These components bring new concepts due to their portability and their ability to be reconfigured. In the beginning, they could not be integrated directly to commercial products, except to make some logical functions. Yet they were widely used in design offices for ASIC prototyping. Their ability to be reconfigured offered a much faster development process to designers.

Now FPGAs are widely used in embedded systems. Indeed, they are used to their multiple advantages:

- they reduce time to market by reducing development duration
- they reduce cost for low sale volume products. This becomes true for sale volume more and more important.
- they permit remote reconfiguration.

Security is becoming a new challenge for all these devices. It is very difficult to estimate the cost of internet fraud. Some talk about hundreds of millions of dollars, others about ten billion dollars. In any case, the experts agree on one point: banking fraud on the internet and fraud on smartphones are steadily increasing. Embedded systems are particularly vulnerable since physical access to these devices is facilitated. Indeed, the owner can be the attacker. [Kocher 2004] explains that security has become an important issue in embedded systems.

Problematic

Confidentiality, integrity and authenticity are often addressed in the literature. However, most contributors address only partially the concept of integrity. Indeed, they forget up-to-dateness that can be affected by a replay attack, especially for FPGA devices.

Updates are a convenient service to fix security flaws. That is why tamper up-to-dateness can be very interesting for an attacker. In fact, a replay attack is a way to downgrade a system in order to exploit vulnerabilities present in a previously fixed version.

Objectives

For this reason, we decided in 2009, in the frame of a collaboration between Netheos company and LIRMM, to conduct research work about the security of embedded systems based on FPGA technologies.

That is why this thesis aims to secure embedded systems based on FPGAs, by increasing the level of security taking up-to-dateness into account.

Contribution

This dissertation proposes robust and industrializable solutions matching requirements and constraints related to embedded systems in an industrial context. By providing dedicated and adapted solutions, this thesis considers the entire life of the embedded system (startup, updates and execution) and the whole system (the FPGA bitstream, the operating system's kernel, code and critical data).

This thesis is distinguished by offering innovative solutions suitabled to the world of FPGAs and providing ways to protect against replay attacks.

Organization

The layout of this document is organized in order to follow step by step how to secure an embedded system to trust it starting from hardware to reach the final software application. That is why the outline of this document begin to address FPGA bitstream to move up through the abstraction layers as follow:

Chapter 1 introduces basic security and cryptography concepts extensively used in this thesis and presents a security taxonomy in order to evaluate the security level of embedded systems. It also introduces specific vocabulary related to cryptography and FPGA to finally present the main attacks targeting FPGAs.

Chapter 2 addresses the securing of both Flash and SRAM (Static Random Access Memory) FPGAs. The first part proposes a secure protocol for remote bitstream update in order to prevent replay attacks on Flash FPGA bitstream. It describes the implementation and evaluates the solution performance and area overhead. The other part presents the SecURe DPR (Secure Update against Replay attacks for DPR (Dynamic Partial Reconfiguration)) protocol. Owing to the fact that DPR is only provided by SRAM FPGAs, SecURe DPR aims to propose a new approach taking into account volatile nature of this reconfigurable device family. The goal is to ensure confidentiality, integrity, authenticity and up-to-dateness of partial bitstreams.

Chapter 3 describes how to secure OSs (Operating Systems) and applications running on previously trusted FPGAs. It presents an overview of currently used counter-measures to protect OSs and applications on FPGAs. It summarizes, organizes and explains existing counter-measures which are not necessarily specific to FPGAs. It also proposes an OS boot verification that allows precluding malicious kernel modifications. This work also ensures kernel up-to-dateness and supports updates management. It highlights the performance improvement of this security mechanism thanks to hardware acceleration mechanisms. It also describes the implementation and evaluates the overhead of this solution in terms of performance and area.

Chapter 4 is an overview of the SecReSoC platform supported and granted by the ANR -French National Research Agency- ARPEGE 2009 program (ANR-09-SEGI-013). It replaces also the work of this thesis into a shared project that aims to offer a complete protection for multiprocessor architectures based on FPGAs.

Finally, a conclusion summarizes the contributions of this dissertation and proposes further works and ideas.

Chapter 1

Background, Issues and Motivations

- Who are you?
- Your worst nightmare.

Zaysen and Rambo - Rambo 3

Contents

This first chapter aims to set out the context of the thesis. It classifies the different types of attack and the tools to evaluate the security level of embedded systems. It also explains basic security principles and introduces the specific vocabulary related to cryptography and FPGA. Finally, it presents the main attacks targeting FPGAs.

1.1 Security and cryptography introduction

Cryptology means *science of secret* and is composed by cryptography and cryptanalysis.

Cryptography is the practice and study of techniques dedicated to keep a message secret. Cryptanalysis is the science to analyze a cryptographic message in order to break its secret. This discipline developed in ancient times was designed for military usage.

Nowadays cryptography would continue to be important for national security and intelligence gathering.

Cryptography provides four main security aspects: confidentiality, authentication, data integrity and non-repudiation. The definitions of this security objectives are taken from [Menezes 1996]:

- **Confidentiality** is a service used to keep the content of information from all but those authorized to have it. Secrecy is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.

- **Data integrity** is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution.

- **Authentication** is a service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons this aspect of cryptography is usually subdivided into two major classes: entity authentication and data origin authentication. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).

- **Non-repudiation** is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute.

This thesis also addresses **up-to-dateness** that can be considered as integrity. Up-to-dateness concept will be described in details into Section 1.2.

1.2 Security technology and metrics

Security and cryptography exist because there are threats and attacks. Section 1.2.1 classifies and describes the different attack types to evaluate the security robustness of a system.

1.2.1 Attacks

Attacks can be distinguished in two families:

- **Passive attacks** which consist in secretly observing, on a communication channel, a private conversation between several parts without their consent (eavesdropping). The main characteristic of a passive attack is to observe without injecting or modify data.
- **Active attacks** which allow modifying messages with deletions, corruptions, injections or replays. In this case, the attacker affects the data flow.

It is possible to separate attacks in two categories considering penetration into the attacked hardware.

- **Non-invasive attacks** which consist in compromising a system *without* hardware intrusion, for instance by probing electromagnetic leakages over a chip, probing an electrical signal on a copper track or by exploiting software vulnerabilities.
- **Invasive or semi-invasive attacks** which consist in compromising a system *with* hardware intrusion, for instance by probing a signal into a chip after removing the chip package with acids and eventually by abrading the die to access inside the silicon.

Non-invasive attacks are particularly dangerous since the user might be not aware of the system compromising. However, this type of attacks does not work directly with the information but with information dependent phenomena. They require a detailed knowledge of both the hardware and software. Considering invasiveness is interesting because it gives an overview of the attacker investment in term of efforts and expenses.

1.2.2 Metrics

This section aims to evaluate the security of a given device.

In [Abraham 1991], IBM (International Business Machines) proposes to classify attackers in three levels according to their strength:

- **Class I (clever outsiders):** They are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.
- **Class II (knowledgeable insiders):** They have substantial specialized technical education and experience. They have varying degrees of understanding of parts of the system but potential access to most of it. They often have highly sophisticated tools and instruments for analysis.
- **Class III (funded organizations):** They are able to assemble teams of specialists with related and complementary skills backed by great funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most advanced analysis tools. They may use Class II adversaries as part of the attack team.

1.2.2.1 Common Criteria

The *Common Criteria for Information Technology Security Evaluation* is best known as CC (Common Criteria). It is an international standard for computer security certification allowing vendors to objectively claim about the security attributes of their products. CC provides assurance that the process of specification, implementation and evaluation of a computer security product has been conducted in a rigorous and standard manner. It is originated out of three standards: ITSEC (Information Technology Security Evaluation Criteria), CTCPEC (Canadian Trusted Computer Product Evaluation Criteria) and TCSEC (Trusted Computer System Evaluation Criteria). These three standards are respectively European, Canadian and Americans.

They propose seven level of certification corresponding to different EALs (Evaluation Assurance Levels). In other words, EAL is a rating describing the depth and rigor of the evaluation.

CC involves five main concepts:

- **TOE (Target Of Evaluation)** is the product or system that is the subject of the evaluation. It can be a set of software, firmware and/or hardware possibly accompanied by guidance.
- **SFRs (Security Functional Requirements)** is a translation of the security functions provided by the TOE into a standardised language. The CC presents a standard catalogue of such functions.
- **PP (Protection profile)** describes the general requirements for a TOE type, a class of security devices.

- **ST (Security Target)** identifies the security properties of the TOE. It may refer to one or more PPs. The TOE is evaluated against the SFRs established in its ST. It is can be built from one or several PPs.

[CC 2009] defines seven EALs.
These levels are summarized in figure 1.1 and detailed below:

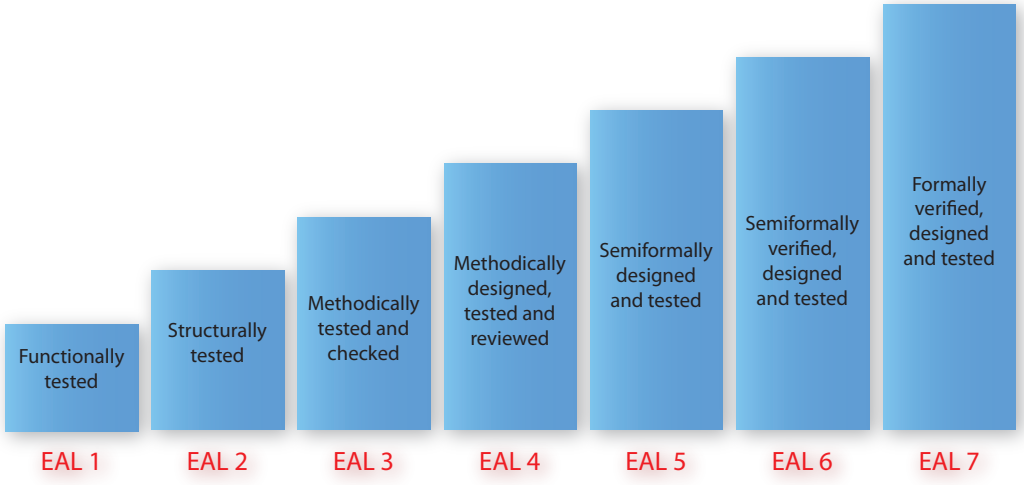


Figure 1.1: The seven EALs of the CC

- **EAL1: functionally tested** EAL1 is applicable where some confidence in correct operation is required, but the threats to security are not viewed as serious. It will be of value where independent assurance is required to support the contention that due care has been exercised with respect to the protection of personal or similar information.

  EAL1 requires only a limited security target. It is sufficient to simply state the SFRs that the TOE must meet, rather than deriving them from threats, OSPs and assumptions through security objectives.

  EAL1 provides an evaluation of the TOE as made available to the customer, including independent testing against a specification, and an examination of the guidance documentation provided. It is intended that an EAL1 evaluation could be successfully conducted without assistance from the developer of the TOE, and for minimal outlay.

  An evaluation at this level should provide evidence that the TOE functions in a manner consistent with its documentation.

- **EAL2: structurally tested** EAL2 requires the co-operation of the developer in terms of the delivery of design information and test results, but should not

demand more effort on the part of the developer than is consistent with good commercial practice. As such it should not require a substantially increased investment of cost or time.

EAL2 is therefore applicable in those circumstances where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record. Such a situation may arise when securing legacy systems, or where access to the developer may be limited.

- **EAL3: methodically tested and checked** EAL3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices.

  EAL3 is applicable in those circumstances where developers or users require a moderate level of independently assured security, and require a thorough investigation of the TOE and its development without substantial reengineering.

- **EAL4: methodically designed, tested, and reviewed** EAL4 permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. EAL4 is the highest level at which it is likely to be economically feasible to retrofit to an existing product line.

  EAL4 is therefore applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity TOEs and are prepared to incur additional security specific engineering costs.

- **EAL5: semiformally designed and tested** EAL5 permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practices supported by moderate application of specialist security engineering techniques. Such a TOE will probably be designed and developed with the intent of achieving EAL5 assurance. It is likely that the additional costs attributable to the EAL5 requirements, relative to rigorous development without the application of specialized techniques, will not be large.

  EAL5 is therefore applicable in those circumstances where developers or users require a high level of independently assured security in a planned development and require a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques.

- **EAL6: semiformally verified design and tested** EAL6 permits developers to gain high assurance from application of security engineering techniques to a rigorous development environment in order to produce a premium TOE for protecting high value assets against significant risks.

  EAL6 is therefore applicable to the development of security TOEs for application in high risk situations where the value of the protected assets justifies the additional costs.

- **EAL7: formally verified design and tested** EAL7 is applicable to the development of security TOEs for application in extremely high risk situations and/or where the high value of the assets justifies the higher costs. Practical application of EAL7 is currently limited to TOEs with tightly focused security functionality that is amenable to extensive formal analysis.

In France, Cofrac (Comité français d'accréditation) accredits CC evaluation facilities, CESTI (Centre d'Evaluation de la Sécurité des Technologies de l'Information), according to norms and standards specified by the ANSSI (Agence nationale de la sécurité des systemes d'information).

In United states, NIST (National Institute of Standards and Technology) accredits CCTLs (Common Criteria Testing Laboratories) according to norms and standards specified by the NSA (National Security Agency).

1.2.2.2 FIPS 140-2

FIPS (Federal Information Processing Standard) 140-2, published by the NIST, defines four levels of security in order to evaluate the level of security offered by cryptographic modules.

Figure 1.2 summarizes the security requirements for the four security levels according to the FIPS 140-2 Standard.

These definitions are taken from [FIPS 2001]:

- **Security Level 1:** provides the lowest level of security. Basic security requirements are specified for a cryptographic module (e.g., at least one Approved algorithm or Approved security function shall be used). No specific physical security mechanisms are required in a Security Level 1 cryptographic module beyond the basic requirement for production-grade components. An example of a Security Level 1 cryptographic module is a personal computer (PC) encryption board.

  Security Level 1 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an unevaluated operating system. Such implementations may be

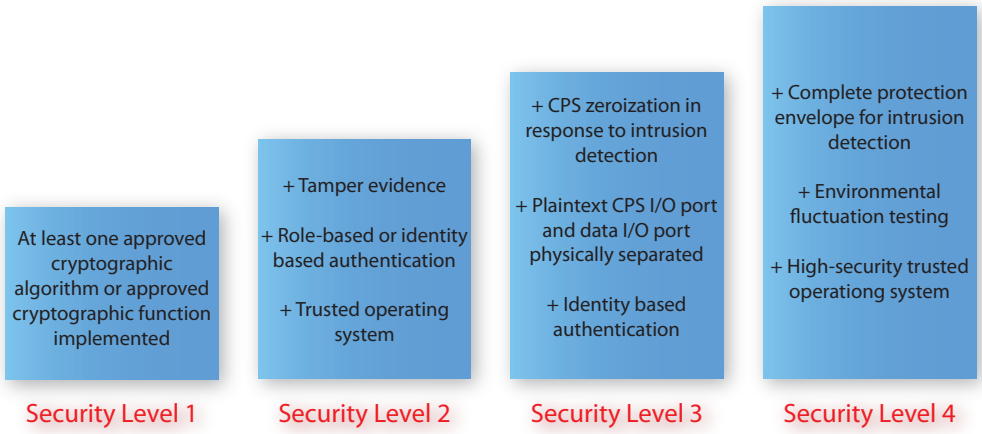


Figure 1.2: The four security levels of the FIPS 140-2 Standard

appropriate for some low-level security applications when other controls, such as physical security, network security, and administrative procedures are limited or nonexistent. The implementation of cryptographic software may be more cost-effective than corresponding hardware-based mechanisms, enabling organizations to select from alternative cryptographic solutions to meet lower-level security requirements.

- **Security Level 2:** enhances the physical security mechanisms of a Security Level 1 cryptographic module by adding the requirement for tamper-evidence, which includes the use of tamper-evident coatings or seals or for pick-resistant locks on removable covers or doors of the module. Tamper-evident coatings or seals are placed on a cryptographic module so that the coating or seal must be broken to attain physical access to the plaintext cryptographic keys and critical security parameters (CSPs) within the module. Tamper-evident seals or pick-resistant locks are placed on covers or doors to protect against unauthorized physical access.

  Security Level 2 requires, at a minimum, role-based authentication in which a cryptographic module authenticates the authorization of an operator to assume a specific role and perform a corresponding set of services.

  Security Level 2 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an operating system that

  - meets the functional requirements specified in the Common Criteria (CC) Protection Profiles (PPs) listed in Annex B and
  - is evaluated at the CC evaluation assurance level EAL2 (or higher).

An equivalent evaluated trusted operating system may be used. A trusted operating system provides a level of trust so that cryptographic modules executing on general purpose computing platforms are comparable to cryptographic modules implemented using dedicated hardware systems.

- **Security Level 3:** In addition to the tamper-evident physical security mechanisms required at Security Level 2, Security Level 3 attempts to prevent the intruder from gaining access to CSPs held within the cryptographic module. Physical security mechanisms required at Security Level 3 are intended to have a high probability of detecting and responding to attempts at physical access, use or modification of the cryptographic module. The physical security mechanisms may include the use of strong enclosures and tamper detection/response circuitry that zeroizes all plaintext CSPs when the removable covers/doors of the cryptographic module are opened.

  Security Level 3 requires identity-based authentication mechanisms, enhancing the security provided by the role-based authentication mechanisms specified for Security Level 2. A cryptographic module authenticates the identity of an operator and verifies that the identified operator is authorized to assume a specific role and perform a corresponding set of services.

  Security Level 3 requires the entry or output of plaintext CSPs (including the entry or output of plaintext CSPs using split knowledge procedure) be performed using ports that are physically separated from other ports, or interfaces that are logically separated using a trusted path from other interfaces. Plaintext CSPs may be entered into or output from the cryptographic module in encrypted form (in which case they may travel through enclosing or intervening systems).

  Security Level 3 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an operating system that

  – meets the functional requirements specified in the PPs listed in Annex B with the additional functional requirement of a Trusted Path (FTP_TRP.1) and
  – is evaluated at the CC evaluation assurance level EAL3 (or higher) with the additional assurance requirement of an Informal Target of Evaluation (TOE) Security Policy Model (ADV_SPM.1).

  An equivalent evaluated trusted operating system may be used. The implementation of a trusted path protects plaintext CSPs and the software and firmware components of the cryptographic module from other untrusted software or firmware that may be executing on the system.

- **Security Level 4:** Security Level 4 provides the highest level of security defined in this standard. At this security level, the physical security mechanisms provide a complete envelope of protection around the cryptographic module with the intent of detecting and responding to all unauthorized attempts at physical access. Penetration of the cryptographic module enclosure from any direction has a very high probability of being detected, resulting in the immediate zeroization of all plaintext CSPs. Security Level 4 cryptographic modules are useful for operation in physically unprotected environments.

  Security Level 4 also protects a cryptographic module against a security compromise due to environmental conditions or fluctuations outside of the module's normal operating ranges for voltage and temperature. Intentional excursions beyond the normal operating ranges may be used by an attacker to thwart a cryptographic module's defenses. A cryptographic module is required to either include special environmental protection features designed to detect fluctuations and zeroize CSPs, or to undergo rigorous environmental failure testing to provide a reasonable assurance that the module will not be affected by fluctuations outside of the normal operating range in a manner that can compromise the security of the module.

  Security Level 4 allows the software and firmware components of a cryptographic module to be executed on a general purpose computing system using an operating system that

  - meets the functional requirements specified for Security Level 3 and
  - is evaluated at the CC evaluation assurance level EAL4 (or higher).

  An equivalent evaluated trusted operating system may be used.

1.3 FPGA and security

The work presented in this thesis deals with FPGAs. That is why these section objectives are to answer some questions.

What is a FPGA?

How the use of FPGAs in the world of security is interesting?

What is the terminology specific to FPGA?

What are the FPGA requirements in terms of security?

1.3.1 FPGAs

A FPGA is an IC (Integrated Circuit) designed, like microprocessor, to be programmed and reprogrammed by the designer after manufacturing. Contrary

to microprocessors, with programmable logic devices, it is possible to change the logical function of the chip. Thus, FPGAs can be used to implement any logical function that an ASIC could perform.

FPGAs contain programmable logic components called *logic blocks*, and reconfigurable interconnects that allow the blocks to be wired together. Logic blocks also include memory elements, which may be basic flip-flops or more complete blocks of memory like block RAMs (Random Access Memories).

Contrary to microprocessor languages which are procedural (or sequential), FPGAs require handling concurrency. In fact, with FPGAs or ASICs, signals propagate simultaneously across all the circuit. It is impossible to describe a circuit with sequential languages which execute instructions the one after the other. That is why FPGA configuration is generally specified using a HDL (Hardware Description Language). Any change to the process's input automatically triggers an update in the simulator's process stack.

The two most used HDL are:

- **VHDL (VHSIC (Very-High-Speed Integrated Circuit) Hardware Description Language):** based on Ada syntax, it was originally developed at the behest of the U.S Department of Defense.
- **Verilog:** based on C syntax, it was a proprietary language of Cadence Design Systems. Since 1995, it is an open standard.

These HDLs have been introduce in 1980s.

Traditionally, FPGAs have been created for prototyping or small volume production. With ASICs, the cost of the first produced chip is very expensive due to the photomask set price. FPGAs permit producing the same chip for several applications.

For more information about FPGAs architecture, refer to this thesis [Ahmed 2011].

1.3.2 FPGA contribution to cryptography

Applications of FPGAs include digital signal processing, software-defined radio, aerospace and defense systems, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas.

FPGAs are commonly used to reach high performance computation due to they offer a full parallel operating mode. In cryptography, this parallelism feature permits hardware accelerators.

If we deal with encryption, although the frequency of FPGAs is lower than that of processors, FPGAs can provide a result (e.g. a 128 bits ciphered output) at each

clock cycle. Processors require a large number of clock cycle to provide the same result and ultimately, FPGAs are faster for this type of processing. Like with ASICs, it is possible to pipeline an architecture implemented in FPGAs.

1.3.3 FPGA terminology

In this document, a FPGA-specific vocabulary is used. It is important to define the terms invoked in the following sections in order to well understand the contribution of this work.

There are four entities implicated in the FPGA life cycle:

- **FPGA vendor** is the entity that designs and produces FPGA chips. The market leaders are Xilinx and Altera. The main other competitors are Lattice Semiconductor and Microsemi (ex-Actel).
- **IPs (Intellectual Properties) designer** designs and provides hardware units often delivered as HDL code or netlist (a list of the logic gates) in the goal to be reused by the system designer. IP cores may be communication controllers or interfaces, specific processors, memory interfaces or encryption engines.
- **SD (System Designer)** designs the system by connecting IPs in order to generate an FPGA-based system.
- **System owner** is the end user that exploits the system. He can often have directly access to the hardware and can be potentially malicious. SDs and IP designers should not necessarily trust him.

The FPGA is composed of two parts:

- **CL (Configuration Logic)** is hard-wired and cannot be configured or modified by the SD. This is this part that is responsible of loading the configuration file (more commonly called *bitstream*) into the user logic. It is composed by at least a configuration port: in most cases the JTAG (Joint Test Action Group) chain (the common name of the *Standard Test Access Port and Boundary-Scan Architecture*). CL can also contain a bitstream decryption engine and/or a bitstream integrity checking mechanism.
- **UL (User Logic)** is the part that can be configured by SD. It can contains LUTs, block RAMs, hardly implemented processors or IPs, DSPs (Digital Signal Processors), and switch matrix for routing signals. When dealing with partial reconfiguration (explained later), UL usually contains two sub-areas:

- **Static area:** the main part of the UL which cannot be partially reconfigured.
- **Partial reconfigurable area:** composed of RPs (Reconfigurable Partitions), in gray in the Figure 1.3. They are considered as slots in UL and are intended to host RMs (Reconfigurable Modules). RPs are reconfigured using partial reconfiguration by sending pieces of bitstream, called partial bitstreams, through the ICAPs (Internal Configuration Access Ports) module.

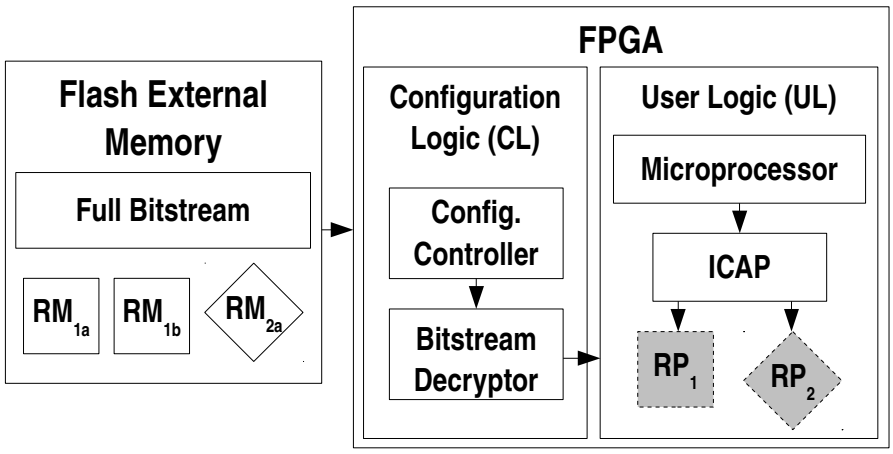


Figure 1.3: FPGA architecture overview

FPGAs are classified in two families:

- **Volatile FPGAs**
  - **SRAM FPGAs:** This technology is the most widespread. These FPGA devices are composed only by SRAM memory cells. It permits having the best performances while using a standard CMOS (Complementary Metal Oxide Semiconductor) manufacturing processes. However they cannot keep their configuration when power is down and need an external non-volatile memory to store the bitstream.
- **Non-volatile FPGAs**
  - **Fuse FPGAs:** Configuration cannot be changed after programming (One-time programmable). Each configuration point is controlled by a fuse element. Before programming, a fuse element has a conductor. Programming consists in permanently break the electrically conductive path (typically with an excessive current). It is the only bipolar technology; others FPGAs use CMOS technologies.

  - **Antifuse FPGAs:** Compared with Fuse, it is the same but the opposite. It is the same because it is a one-time programmable device. But it is the opposite because, in addition to be a CMOS technology, an antifuse element is an insulator before programming. Programming consists in permanently create an electrically conductive path (typically with an excessive voltage).
  - **EPROM (Erasable Programmable Read-Only Memory) FPGAs:** This technology is an array of floating-gate transistors and is one-time programmable but can be erased with strong UV (UltraViolet) light. They are easily recognizable by a window in the top of the package, through which the silicon chip is visible. This window permits exposure to UV light during erasing.
  - **EEPROM (Electrically Erasable Programmable Read-Only Memory) FPGAs:** They have the same characteristic than EPROM FPGAs but they are design to be erased electrically instead of with UV light. That is why there is no window in the top of the package.
  - **Flash FPGAs:** They are EEPROMs with a greater storage density (ie with a lower cost). This is the current and most used non-volatile technology. The configuration memory is distributed in the component in order to have a truly single chip FPGA, completely live at power-up, with no internal loading of configuration. This technology has lower performances than SRAM FPGAs but offers a higher security level.

There are several types of reconfiguration:

- **Entire reconfiguration:** it is the "normal" way to reconfigure a FPGA. It consists in substituting the previous bitstream by the new one.
- **Partial reconfiguration:** it is the process of configuring only a portion of a FPGA. In this case, partial reconfiguration can be divided into two groups:
  - **Static partial reconfiguration:** In this case, the device is stopped during the reconfiguration process. While the partial data (or partial bitstream) is sent to the FPGA, the rest of the device is stopped and brought up after the configuration is completed.
  - **Dynamic partial reconfiguration:** It permits changing the part of the FPGA while the rest of the architecture is still running. It can be interesting for applications where it is important to never be unavailable.

We distinguish two partial reconfiguration cases:

- **Module-based partial reconfiguration** permits the reconfiguration of a predefined part of the architecture. In this case each version of the module requires having the same inputs and outputs. The module size must be well appreciated during the design specification because it is impossible to put a too large module in a too small slot.

- **Difference-based partial reconfiguration:** It consists in reconfiguring only changes between two bitstreams. The partial bitstream contains only information about these differences. It can be used when the design is affected by small changes. It is often used to change the content of a dedicated memory blocks.

1.3.4 Conclusion

The reconfigurable aspect may be considered as a drawback for security-sensitive applications. For example an attacker could easily modify the design. That is why certain types of FPGA are one-time programmable.

But the reconfiguration feature can be an advantage. The fact that a SD can remotely reconfigure an FPGA permits him to apply security updates in order to fix vulnerabilities. In this case remote update for hardware systems is a convenient service, impossible with ASICs but enabled by FPGAs. That turns out to be essential when high volume sale products are considered. Effectively it would be extremely expensive or impossible for a constructor to recall a large number of devices.

High performances, low cost, and remote updates must be the main reasons to use FPGAs in security applications, but in this case, security concerns should be taken into account.

1.4 Threats targeting FPGAs

1.4.1 Side channels attacks

Side channels attacks are a large family of cryptanalytic techniques that exploit unexpected properties of a system due to its software or hardware implementation. A side channel can be defined by an information leakage that is not required by the system to complete the functionality. Indeed, a mathematical security does not necessarily guarantee security when using in practice. In this area, the attacks are numerous and focus on different parameters.

1.4.1.1 Timing attacks

This attack analyzes the time taken to perform certain cryptographic operations in order to discover secret information and can be remotely performed. The basis of this attack is based on the fact that, for the purpose of optimizing the software implementation, some operations of cryptographic algorithms running in a non-constant time. These optimizations include, for example, avoidance of unnecessary operation or use of memory cache *(cache attack)*.

For example, in RSA (Rivest, Shamir and Adleman) algorithm, the execution time for the square-and-multiply algorithm used in modular exponentiation depends on the key. The *square-and-multiply* algorithm is executed for each bit of the key: a "1" requires two computations while a "0" requires only one computation. Thus, it is possible to retrieve the key value thanks to the execution time.

The first timing analysis has been described by Kocher in [Kocher 1996].

1.4.1.2 Power analysis

Power or consumption analysis consists in studying the power consumption of a system in order to discover secret information such as encryption key. The energy consumed by a chip is highly dependent on the process running. Some operations, more expensive, increase the power consumption of the circuit. This analysis of variations and spikes permits extracting valuable information to the cryptanalyst.

SPA

SPA (Simple Power Analysis) is the simplest attack based on power consumption and introduced by Kocher in [Kocher 1998]. It consists in a visual examination of a single current trace acquisition over the time. Each microprocessor instruction or device operation involves a different number of transistors. At any time, measuring the current consumption may reflect the activity of the device or the microprocessor.

Figure 1.4, from [Wikipedia], shows a SPA trace permitting to decode RSA key bits thanks to the square-and-multiply operations. The left peak represents the CPU power variations during the step of the algorithm without multiplication, the right (broader) peak represents the CPU power variations during the step of the algorithm with multiplication. This analysis allows reading bits 0, 1.

Information gathered during SPA can be used to improve other more complex power attacks.

DPA

DPA (Differential Power Analysis), presented by Kocher in [Kocher 1998] and

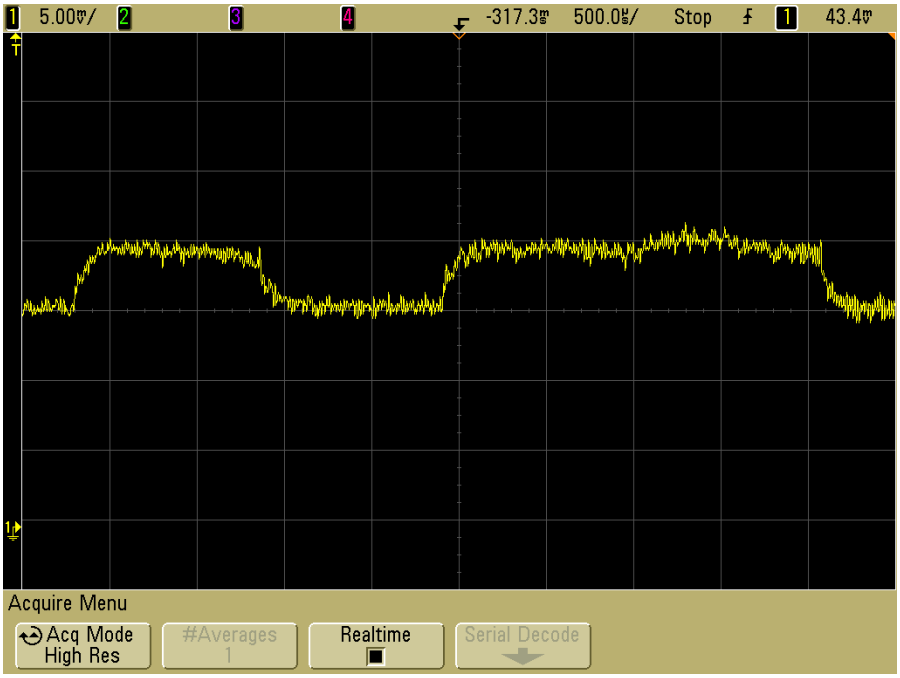


Figure 1.4: SPA trace allowing to read bits 0, 1

[Kocher 1999], combines the analysis of consumption with statistical methods for several operations. It does not require any knowledge about the algorithm implementation.

The attack by analysis of differential consumption is more powerful than SPA attack because it requires less information on the implementation of the attacked algorithm.

HO-DPA (High-Order Differential Power Analysis) [Messerges 2000], an advanced type of DPA attack, involves multiple data sources and incorporates time offsets in the attack. Although the HO-DPA is less widely practiced than SPA and DPA, it makes more devices vulnerable.

1.4.1.3 EMA

EMA (ElectroMagnetic Analysis) is the analysis of electromagnetic radiations emitted by integrated circuits. These radiations are mainly due to the moving of loads through power and ground network (the rails of the metal layers). Thus, transistors switching causes the moving of loads that creates variations of electromagnetic radiations. These radiations are closely correlated to the chip activity. Quisquater presents EMA in [Quisquater 2001]. This analysis can be done with a specific equipment (refer to Figure 1.5) on a whole chip (refer to Figure 1.6(a)) similarly to power analysis or with a smaller sensor (refer to Figure 1.6(b)) allowing

to focus on small portion of the chip. Thanks to that, it is possible to perform cartography.

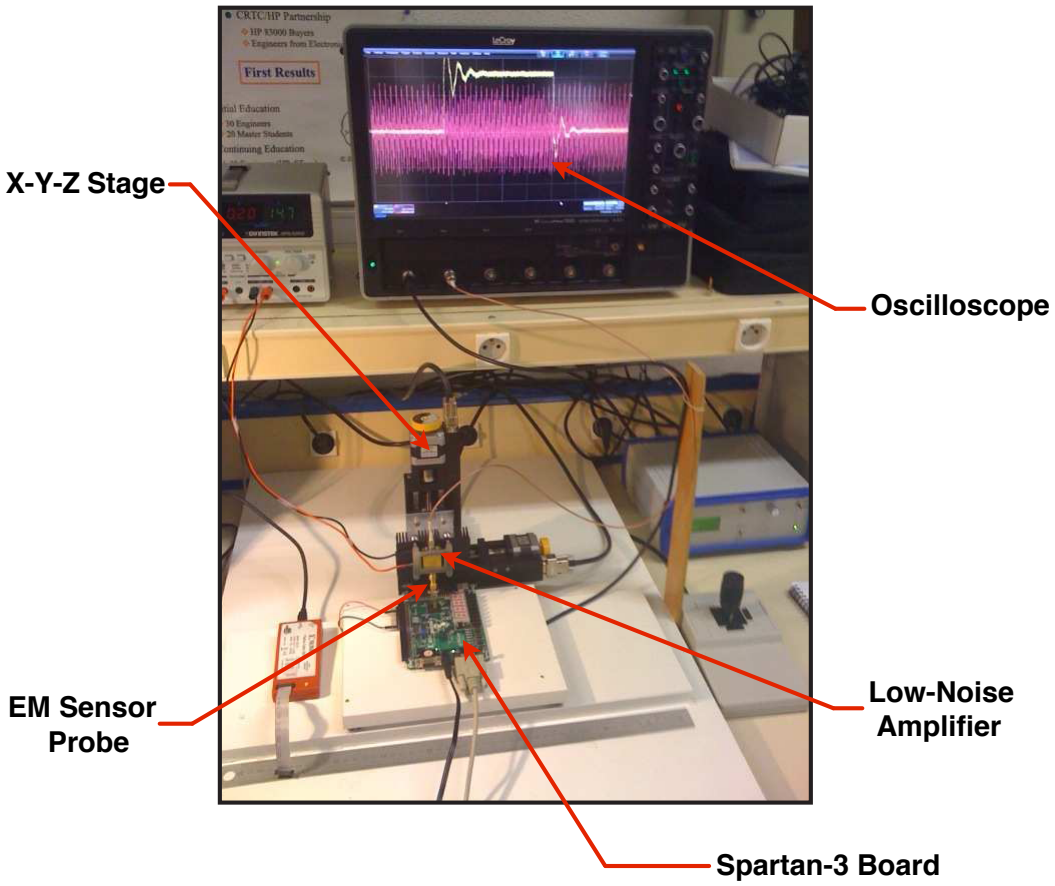


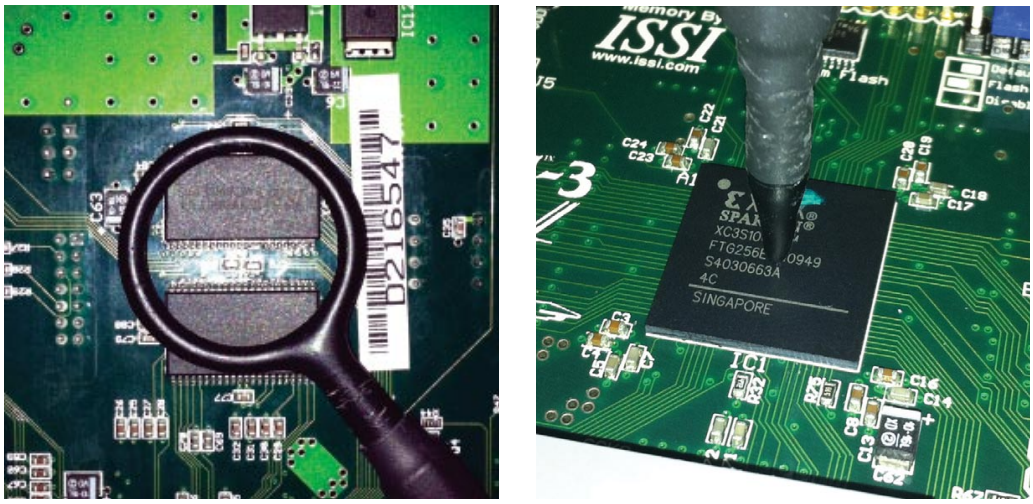
Figure 1.5: Lab equipments involved in electromagnetic analyses

EMA is very similar to power analysis. It is possible to process to SEMA (Simple ElectroMagnetic Analysis) like SPA or to DEMA (Differential ElectroMagnetic Analysis) like DPA. It also exists CEMA (Correlation ElectroMagnetic Analysis) and TEMA (Template ElectroMagnetic Analysis).

1.4.1.4 Acoustic cryptanalysis

This cryptanalysis consists in analyzing the acoustic sounds of the equipment performing cryptographic operations.

The first acoustic cryptanalysis has been done during the cold war. The British intelligence agency named Security Service but commonly known as MI5 (Military Intelligence, Section 5) spied the Egyptian embassy thanks to a microphone placed near the rotor of the Egyptian Hagelin cipher machine. MI5 was able to recover



(a) A global electromagnetic analysis sensor (b) A localized electromagnetic analysis sensor

Figure 1.6: Timing overhead

some plaintext by analysis click sounds produced by the machine.

Emanations analyses are also named TEMPEST attacks. TEMPEST is a codename used by the NSA referring to studies and investigations about electromagnetic leaks. Wim van Eck publishes the first analysis of the security risks of emanations from computer monitors and presents a real attack with low cost equipment in [Van Eck 1985].

In [Shamir 2004] Shamir and Tromer adapt this attack to microprocessors.

1.4.2 Data remanence

Data remanence is the property of residual data that persists even after attempting to remove or delete them. That can be caused by software or hardware.

For instance, software issues can be data being left intact on a non-volatile support due to a file system deletion operation that not really erase the data on hardware.

Physical properties of the hardware storage medium that allow retrieving data considered erased. [Halderman 2009] presents a *cold boot attack*. Security processors or IPs store their key in volatile memories. When the device is turned off, possibly due to violation detection, these volatile memories lose power. At room temperature data are promptly erased but with a temperature of -20 ° C the contents of SRAM can be "frozen". Thus the data and possibly the keys remain readable for several minutes. 1.7 shows a cold boot attack.

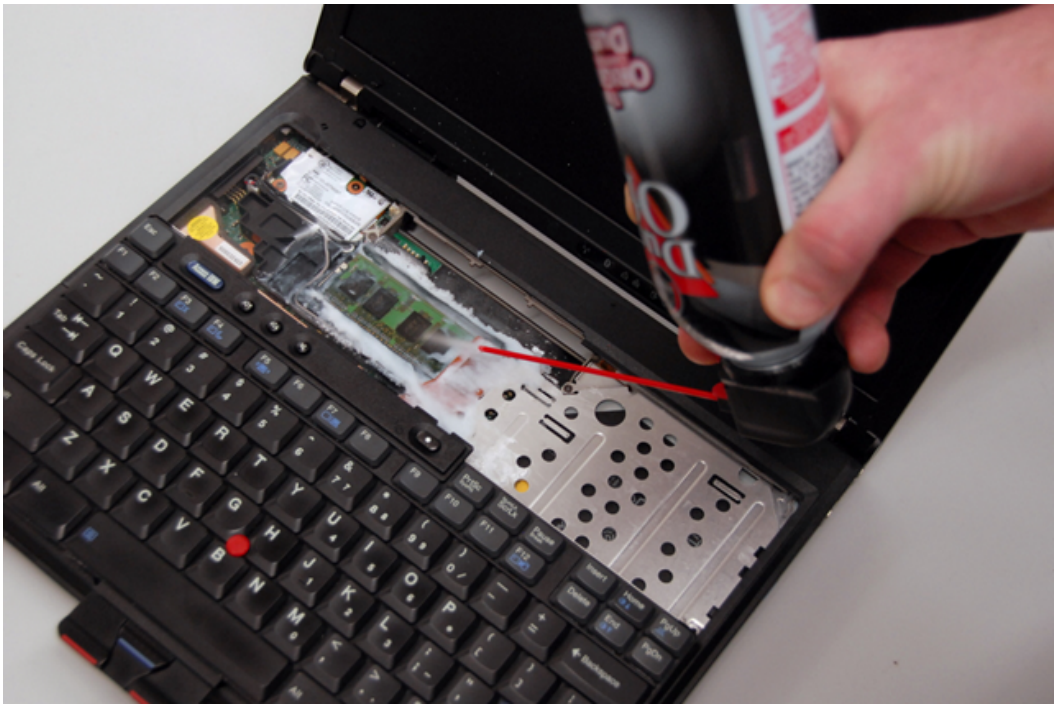


Figure 1.7: A cold boot attack on a computer RAM (from Princeton University)

1.4.3 Probing

The principle of a probing attack is to spy the electrical activity of an electronic component or the communication between two components. 1.8 shows a probing procedure.

Probing permits an attacker to spy communications and to insert chosen texts (active attack). That offers the possibility to perform four kinds of attacks.

- **Read back attack:** the attacker can insert instruction, for instance in destination of the memory controller, in order dump all the memory.
- **Spoofing attack:** the adversary replaces the data transmitted over the bus by his own.
- **Splicing attack:** this is a spatial permutation of memory blocks: the attacker relocalizes a memory block. This is possible, for example, by spoofing the destination address transiting on the bus.
- **Replay attack:** this is a temporal modification. First, the adversary records a data transfer. Later, it *replays* this transfer through the network.

It is possible to perform such attack even if the data are encrypted.

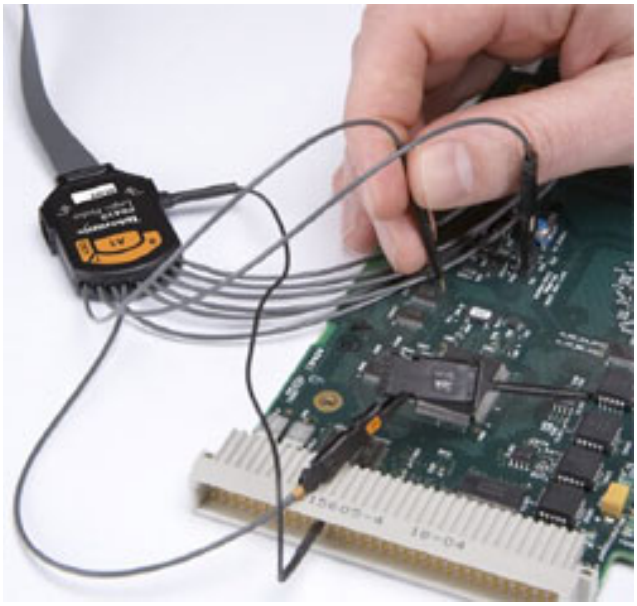


Figure 1.8: A probing procedure

It is used for the extraction of information, such as the contents of the memory targeting secret keys. In [Huang 2003], Huang explains his attack on the X-Box (the Microsoft game console), using a probing attack. This game console has been hacked by analyzing and modifying the processor data and instructions during the execution.

1.4.4 Fault injection

Fault injections are a family of techniques which are to produce voluntarily errors in a cryptosystem. Their purpose is to cause unexpected behavior of the cryptographic operations in order to extract secret information (such as an encryption key).

A fault injection can be performed at every level. The only assumption is that the attacker can affect the internal state of the system by writing values. It can be performed for instance by:

- probing a bus between a processor and the memory or directly into a chip;
- lighting a targeted transistor or memory block with a laser;
- heating the chip in order to create setup/hold time violation (temperature affects the speed of signal propagation);
- increasing the frequency to produce the same effect as heating the chip;
- increasing or decreasing the power supply voltage.

1.5 Conclusion

Embedded systems have expanded significantly in recent years thanks to their growing performances. FPGAs are essential devices in embedded systems thanks to their reconfigurable features.

Up to now, FPGAs' capabilities and threats have been presented, and it appears a significant problematic: if such systems are technologically, economically and commercially major players, it is essential to protect them against a considerable number of attacks while avoiding significantly impacting performances to remain consistent with the expectations of embedded systems.

The next sections propose contributions to enhance this situation, considering the entire lifecycle and addressing the two types of FPGAs which are widely used in embedded systems: Flash-based and SRAM-based FPGAs. This work considers the FPGA in the embedded system addressing both basic and advanced FPGA capabilities.

Chapter 2

Bitstream Security

- I sense a trap.
- Next move?
- Spring the trap.

Obi-Wan Kenobi and Anakin Skywalker -
Revenge of the Sith, Star Wars 3

Contents

This chapter addresses the securing of both Flash and SRAM FPGAs. The first part proposes a secure protocol for remote bitstream update in order to prevent replay attacks on Flash FPGA bitstream. It describes the implementation and evaluates the solution performance and area overhead. The other part presents the SecURe DPR protocol. Owing to the fact that DPR is only provided by SRAM FPGAs, SecURe DPR aims to propose a new approach taking into account volatile nature of this reconfigurable device family. The goal is to ensure confidentiality, integrity, authenticity and up-to-dateness of partial bitstreams.

2.1 Introduction

Remote update for hardware systems is a convenient service enabled by FPGA based systems. This service turns out to be essential. Indeed, in high volume sale products (like set-top boxes) or space-based systems it is too expensive or impossible to retrieve the device in order to update it.

However, the remote feature allows a set of attacks that may challenge the confidentiality and the integrity of the FPGA configuration: the bitstream. For example transmissions of hardware IPs through an insecure network is an important issue. An attacker can record the transmitted bitstream in order to steal the whole bitstream or some IP blocks from an IP vendor.

He can also downgrade the system in replaying a previous version with known flaws. That is a good way to exploit previously fixed vulnerabilities. Indeed updates are the only possibility for a designer to protect his system against threat progress.

Several security schemes providing encryption and integrity checking of the bitstream have been proposed in the literature. However, they do not detect the replay of old FPGA configurations.

This chapter is composed by two parts in order to propose solutions for Flash FPGAs in Section 2.2 and for SRAM FPGAs in Section 2.3.

2.2 Flash FPGAs

Considering FPGAs with embedded NVM (Non-Volatile-Memory), this section proposes a new protocol ensuring bitstream confidentiality, integrity and preventing old bitstreams replay.

That is why, this work proposes a new protocol focusing on spoofing and replay attacks (refer to Section 1.4.3) in order to guarantee bitstream confidentiality, integrity and to prevent old bitstreams replay.

In this work, previous presented ideas [Badrignans 2008] are improved and implemented in order to achieve more flexibility. That is why this contribution insists on the way to manage bitstream versions. This work also highlights the low area overhead and the almost zero performance overhead of our solution.

2.2.1 Threat model

The threat model assumes that the FPGA system is exposed to hostile environment where physical but non-invasive attacks are feasible. FPGA chip is considered as a trusted zone. Typically, side channel attacks on the FPGA device are not considered. For DoS (Denial of Service) considerations, destruction and power off attacks are not considered. Only remote DoS are considered.

Only non-volatile FPGAs including an on chip user NVM are considered. This point will be discussed later on.

All attacks that allow reading, modifying or replaying the bitstream directly on the board or through the network are considered. The SD is supposed trusted and the FPGA platform is initialized in a trusted area.

Typically in "man-in-the-middle" attacks, an attacker is assumed to be placed upon the network between the SD and the FPGA. He is able to retrieve and modify all the communications transmitted through this network. Considering such an attacker, this work focuses on spoofing and replay attacks.

Figure 2.1 describes a replay attack affecting the FPGA bitstream. It details the three steps of such an attack:

1. The SD sends the version i to the FPGA. Meanwhile the adversary records the data transfer.

2. The SD updates the FPGA to a version $i+n$. Here the FPGA is well updated.

3. The adversary replays the data transfer recorded in step 1. This replay can be also performed by substituting the data transfer of the version $i+n$ by the data transfer of the version i. By this way the FPGA remains to the same version instead of being upgraded then downgraded. In all cases, the goal of the attacker is achieved: back to the version i.

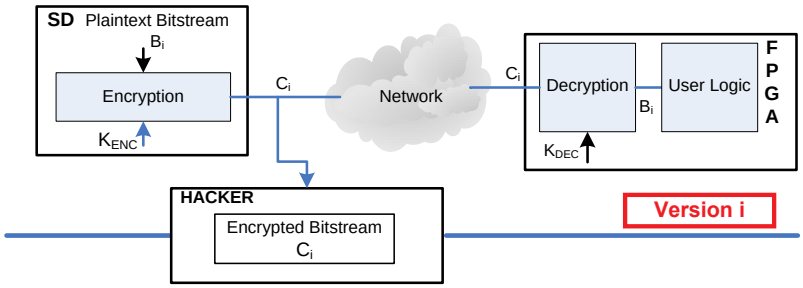
Replay attacks are particularly dangerous, because, as explained in Section 2.2.2.1, the current solutions proposed by FPGA vendors to ensure bitstream encryption and integrity are inefficient against replay. It is possible to downgrade a system in order to exploit its vulnerabilities present in a previous version. An update may typically be performed to correct a critical security flaw.

2.2.2 Related works

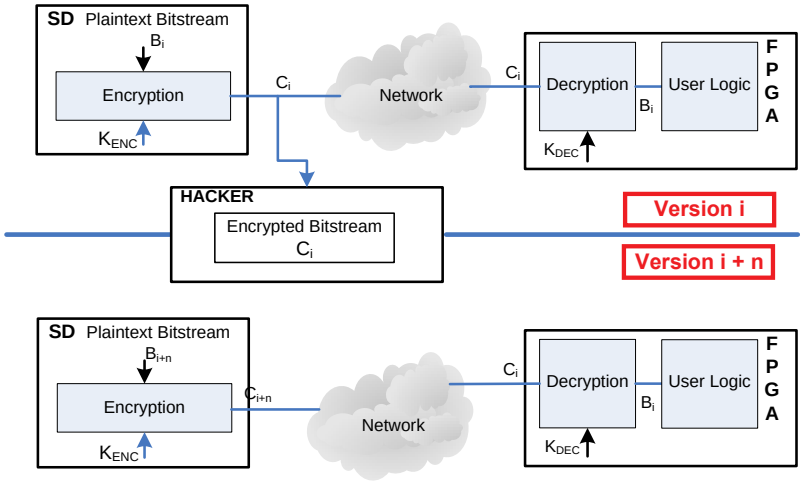
2.2.2.1 Bitstream confidentiality and integrity

Confidentiality ensures that the bitstream can only be read by the cryptographic key owners. This feature is available in a large number of FPGAs. It is generally performed thanks to a hard-wired block cipher cryptocore (for instance AES (Advanced Encryption Standard) [Altera] or 3-DES (Triple Data Encryption Standard) in [Microsemi 2010b]).

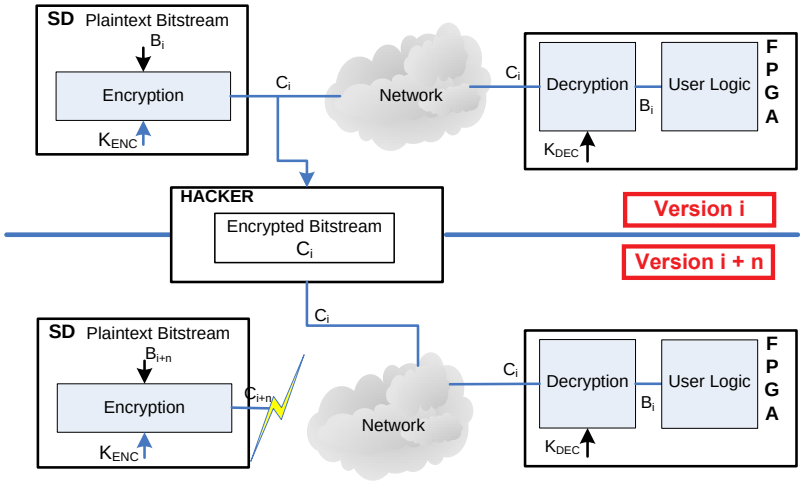
Integrity mechanisms protect the system against unauthorized modifications. It ensures that the bitstream does not undergo any fortuitous or malicious modifications. Some FPGA vendors provide a CRC (Cyclic Redundancy Check) [Höflich 1994]. In this case the device is only protected against fortuitous modifications because CRC is proven as not cryptographically resistant



(a) Step 1



(b) Step 2



(c) Step 3

Figure 2.1: The three steps of a replay attack

[Chawla 2009]. Indeed, CRC codes are exposed to collisions and can be relatively easily broken with brute force attacks since they are only 16 or 32-bit long.

To efficiently protect the data integrity against unauthorized modifications of a bitstream fragment, some FPGA vendors like Microsemi with the ISP (In-System Programming) system [Microsemi 2009] use AES-based MAC (Message Authentication Code).

Currently, no FPGA vendors provide any integrity mechanism that can prevent whole bitstream modifications, for instance substitution by an older one in case of replay attack.

2.2.2.2 Secure remote update preventing replay attacks

Most FPGA vendors provide solutions to facilitate remote update management but do not prevent replay attacks. For instance Altera provides an IP core called altremote update megafunction [Altera 2009], that allows the designer to easily provide bug fixes without product recall, and by this way, extends device lifetime. However Altera does not provide solutions to perform this process securely, especially mechanisms preventing the replay attack.

Conceptually, integrity ensures bitstream up-to-dateness since it avoids data modifications. In fact, replacing a bitstream by an older one is considered as a data modification. But, integrity checking mechanisms, like hash algorithms or MACs, do not ensure up-to-dateness without the use of a temporal reference. In practice, none of them are designed to handle these versioning issues. Nevertheless, academic literature proposes mechanisms against downgrades.

In [Drimer 2009], Drimer describes a system with several flash memory slots. In case of update failure due, for example, to a power outage during an update process, the FPGA platform can start thanks to a bitstream present in a rescue slot. This secure mechanism requires involving an embedded processor and was never implemented and tested to our knowledge.

Braeken et al. propose STRES (Secure Techniques for Remote reconfiguration of Embedded Systems) [Braeken 2011] based on STS (Station-to-Station) protocol also requiring processor instantiation. It uses elliptic curve communicating through a TCP/IP (Transmission Control Protocol / Internet Protocol) connection.

In [Badrignans 2009], Badrignans describes three mechanisms:

1. The first one is applicable to any FPGA device supporting bitstream encryption and integrity checking. It is based on regular challenge-response verification. This protocol needs a very constraining regular polling.

2. The second one is applicable to any non-volatile FPGA embedding a user NVM. This user NVM allows storing the current bitstream version number.

At each FPGA boot this value is compared to bitstream version to ensure its up-to-dateness.

This solution requires having a specific bitstream for each FPGA that considerably complicates the bitstream management of wide FPGA sets.

3. The last one requires modifying the static logic (the hard-wired FPGA part) and can be only set up by FPGA vendors.

2.2.3 Secure update principle

The goal of this secure update mechanism is to lock the FPGA to a specific version in order to prevent replay attacks.

This contribution is an extension of the work presented in the second mechanism of [Badrignans 2009] in view to correct this bitstream specificity issue. It also improves this protocol to be resistant to update failures.

2.2.3.1 Generic design overview

The figure 2.2 shows the FPGA design enabling to implement the secure remote update of bitstream mechanism. The FPGA is composed of three parts.

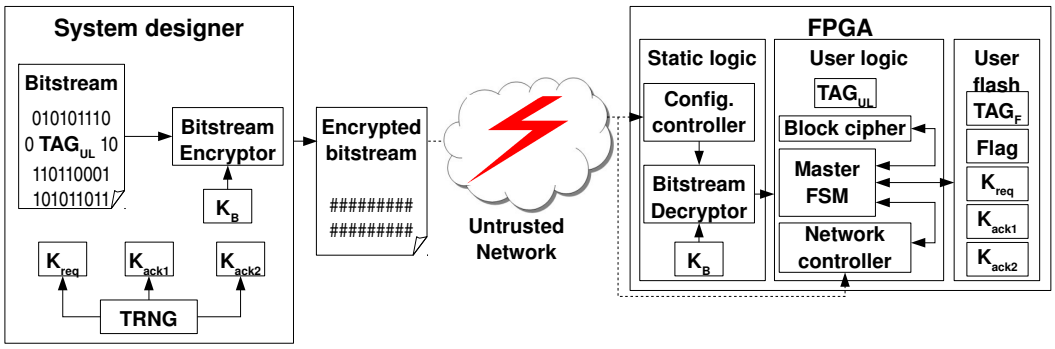


Figure 2.2: Design overview

1. The first one, static logic, is hard-wired and cannot be configured. It contains a decipher, protecting the bitstream confidentiality and integrity, and its key. This key, named K_B, is only known by the FPGA and the SD.

2. The second one, user logic, can be configured by the SD and contains a bitstream version verification mechanism. It is composed of a FSM (Finite State Machine) able to manage:

   - A network controller: to enable the update to be remotely performed.

- A Non-volatile memory controller: to store a first power-up flag, the current bitstream version number and keys shared with the SD.
- A block cipher: to ensure the mutual authentication of the FPGA and the SD.

3. The third one, user NVM contains three keys shared with the SD. They must be unique for each FPGA and are used to encrypt the tag:
   - K_{req}: for the Update command.
   - K_{ack1}: for the Update command acknowledgment.
   - K_{ack2}: for the new bitstream version reception and start-up on the good device acknowledgment.

Since the goal is to lock the FPGA to a particular version, the NVM contains also the value indicating the current one. This value, named TAG_F can be only incremented by the SD. It will be compared to the tag contained in the user logic, named TAG_{UL} (refer to figure 2.2). Each Bitstream version contains his proper TAG_{UL}.

In practice, it is a constant in the design source code: version zero is tagged with number zero, version one with number one, and so on. The NVM is written the first time from outside of FPGA chip in a trusted zone before to be locked using FPGA vendor mechanism [Microsemi 2009]. After locking, the NVM can be read and write only from the user logic.

2.2.3.2 Update process

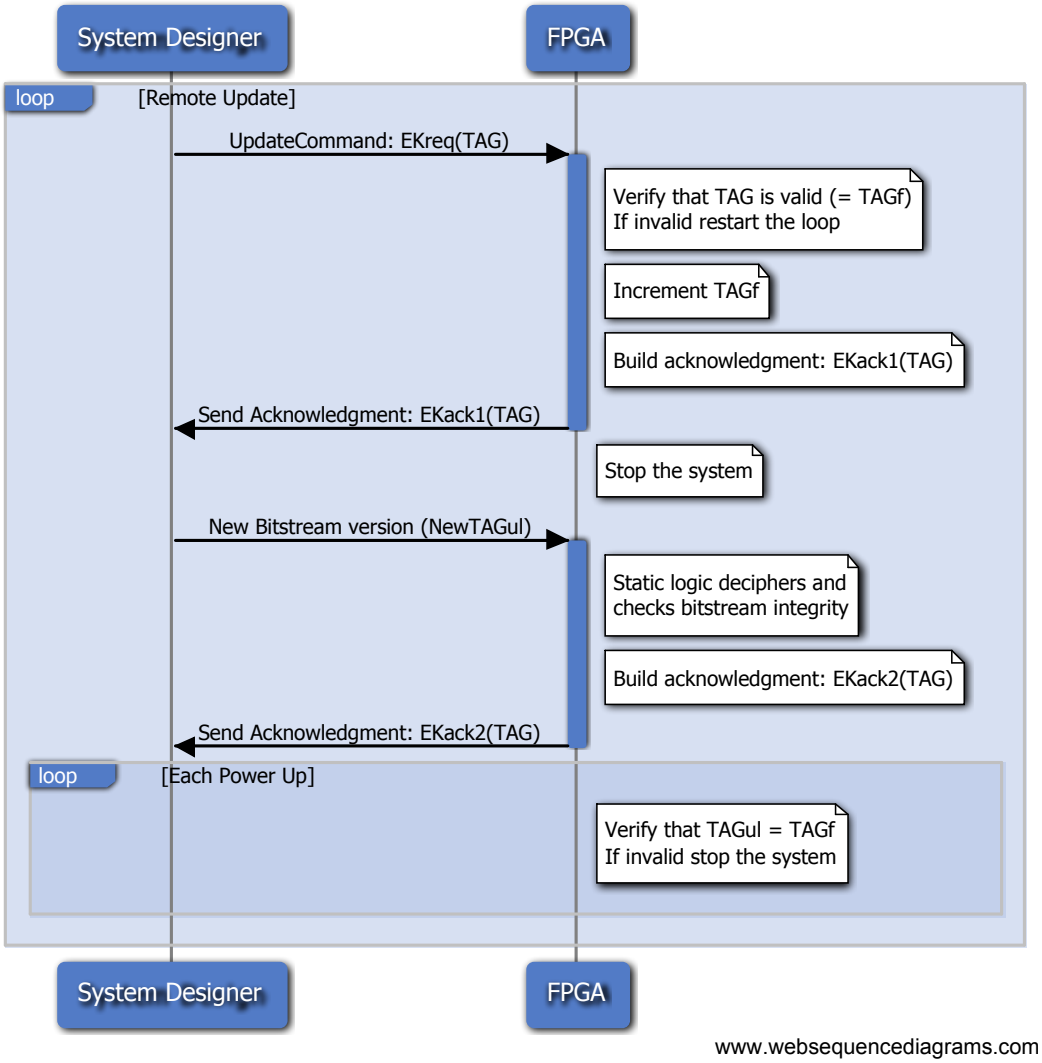
The figure 2.3 focuses on communications between the SD and the FPGA. It explains the process able to verify that the current bitstream version is genuine and increment securely this non volatile value in view to a future update.

This update process is described more precisely in the following section.

Update command

This command increments TAG_F with a view to prepare the FPGA to an update. The SD sends the update command containing the tag encrypted with the K_{req} as cipher key to the FPGA. This encryption is required to prevent that an attacker could increment this TAG himself. Cipher mode can be non-chained if TAG_F can be contained in one block cipher input.

After decryption, the FPGA compares the tag contained in his own bitstream (TAG_{UL}) and the tag sent by the supposed SD. If they are different, the FPGA continues to work and waits for a new update command. Else it is sure that the



EKx(M) : M encrypted with K_x as cipher key.

Figure 2.3: Remote secure update protocol diagram

command comes from the SD. In this case, the FPGA increments TAG_F and starts to encrypt the new tag with the cipher key K_{ack1}. To inform the SD that the tag incrementing command is received, the FPGA sends the result of the encryption. Thereby the SD, who knows the K_{ack1}, can check the FPGA has received the update command and the new TAG_F is effectively up to date. Now, the FPGA is ready to receive the bitstream, and so, to become up-to-date. The design is stopped.

Download a new bitstream version

When the previous step is validated, the SD sends the new ciphered bitstream, with its MAC, to the FPGA.This protects the FPGA against malicious bitstreams.

At the start-up of this new design, the FPGA performs the new tag encryption with K_{ack2} as cipher key and sends the result to the SD.

This acknowledgment informs the SD that the new bitstream was correctly downloaded to the right FPGA and the design was effectively started-up on the right FPGA. In fact the new TAG is contained only in this new bitstream and the K_{ack2} is known only by this FPGA.

Bitstream version verification

Each time the FPGA design starts-up, it will check that TAG_{UL} and TAG_F are equals, using a comparator as wide as TAGs length. If they were different, a replay attack is detected, an alarm (a signal in the design) is triggered that can be used by the SD to apply his policy. He can for instance stop, destroy the system or enter in a degraded mode.

2.2.4 Security analysis

This analysis focuses on bitstream replay and remote DoS attacks.

Our scheme assumes that the FPGA vendors encryption and integrity verification mechanisms (refer to Section 2.2.2.1) are secure. For instance, as it is explained in [Microsemi 2010b], ISP Microsemi mechanism, implemented in the static logic, checks the bitstream integrity thanks to a MAC while the device is still operating. If the MAC validates the bitstream, the device will be erased and programmed. Else, the device will continue to operate uninterrupted and does not take the new bitstream into account. The new bitstream tag cannot be changed by an attacker thanks to the MAC.

This should preclude DoSs in the case that an attacker modifies the end of the bitstream. Actually, contrary to the documentation [Microsemi 2010b], the device is not erased and programmed after checking the integrity of the whole bitstream. The buffer used to store the bitstream during the verification is not large enough. By this way, the ISP mechanism splits the bitstream in several small sized fragment in order to verify and and copy it to the FPGA, the one and then the other. That is why, if an attacker modifies only the end of the bitstream without modifying the begining, the ISP start de overwrite the right bitstream.

In a nutshell, this update protocol is not DoS resistant, contrary to what the Microsemi documentation [Microsemi 2010b] says.

The tag is encrypted with three different keys to prevent replay attacks. Indeed, to avoid that the attacker responds by pretending to be the device or the SD, only one-time messages (key-tag pairs) are transmitted upon the untrusted network. If it is not the case, everyone can replay this message to increment the TAG in order to perform a remote DoS attack. Since, for a bitstream version, the tag is the same for all the FPGAs, K_{req}, K_{ack1} and K_{ack2} must be unique for each device.

In the step, which is to download a new bitstream version, the new TAG_{UL} cannot be spoofed because his integrity is verified thanks to a MAC (cf. Section 2.2.2.1). For the same reasons an attacker cannot replace this bitstream by his own.

The bitstream first boot acknowledgment able to detect update failures. If the SD does not received this acknowledgment, he can send the new bitstream until the FPGA succeed to boot on him.

2.2.5 Implementation considerations

This new secure remote update of bitstream protocol can be implemented on other FPGA platforms. The only requirement is the presence of:

- An embedded flash memory in the FPGA chip to securely store the first power-up flag, the FPGA current TAG, K_{req}, K_{ack1} and K_{ack2}.
- A mechanism proposed by the FPGA vendor providing bitstream confidentiality and integrity.

For instance Xilinx with the Spartan3-AN FPGA [Xilinx 2009] embeds a flash memory. It also provides a DNA mechanism, explained by Xilinx in [Smerdon 2008], but that does not protect bitstream confidentiality.

It is possible to implement a cryptocore in user logic to decipher the encrypted bitstream provided by the SD before storing it in configuration flash. Bitstream integrity must be also verified with a secure mechanism based for instance on a MAC. This securing needs an extra cost in terms of area.

Spartan3-AN provides several bitstream slots and allows a better robustness against update failures with a Drimer-like mechanism like it is explained in [Drimer 2009].

Lattice provides also such services in XP2 [Lattice corporation 2010a] and MachXO2[Lattice corporation 2010b] FPGA Family. Bitstream confidentiality and integrity are provided but NVM is used to save a RAM copy and it is less easy to store data: the first power-up flag, TAG_F, K_{req}, K_{ack1} and K_{ack2} in our case. A rescue bitstream slot, named "golden", is also provided by Lattice, protecting against update failures.

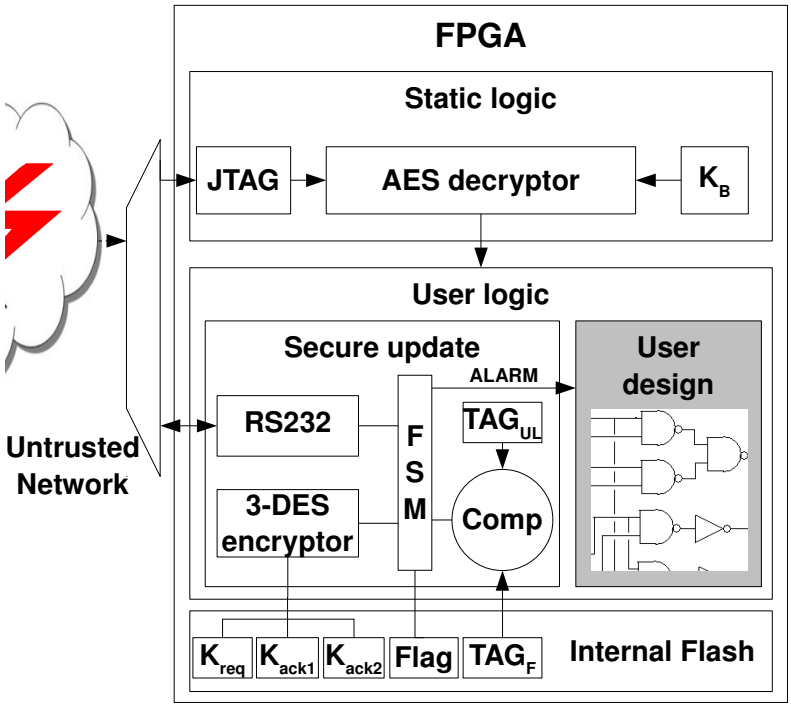


Figure 2.4: FPGA-focused overview

2.2.5.1 Demonstration platform

A demonstrator has been developed to validate the protocol and to evaluate the performance and area overhead of such a mechanism. This demonstration platform is based on an Microsemi Fusion starter kit (FPGA: Fusion AFS600). It is a non-volatile FPGA, embedding a user flash memory and providing a confidentiality and integrity mechanism. Figure 2.4 describes this demonstrator.

The network input is composed of a JTAG port and a serial communication port based on RS232 link:

- The first one (JTAG) is used to received the bitstream.
- The second one (RS232) is used to receive update commands and send the two acknowledgments (the first-boot and the tag incrementing ones).

In a real application it shall be replaced by a network adapter.

The block cipher used in this implementation is a 3-DES. AES uses about two times more tiles than 3-DES and needs a more complex design on Microsemi FPGAs because of block RAMs initialization.

The fact that K_{req}, K_{ack1} and K_{ack2} are stored in the flash memory with TAG_F is very useful for applications mostly when a large number of FPGAs are managed and of course avoiding in this way a vulnerability at the first FPGA initialization. This

allows producing the same bitstream file for the whole set of FPGAs contrary to the work presented in [Badrignans 2008], since, for a given version of the bitstream that is only the three keys, K_{req}, K_{ack1} and K_{ack2}, that differentiate the device.

Indeed during the initialization procedure this three keys and TAG_F are downloaded through the JTAG port before locking. Then the flash access through this port is locked. This locking prevents that an attacker reads or overwrites the keys and the TAG_F, otherwise the whole protocol will be compromised.

The figure 2.5 describes the FPGA Master FSM algorithm.

The Section 2.2.3.2 explains how the SD is authenticated by the FPGA. The K_{req}-ciphered tag is received from the supposed SD. After decryption, a comparison between the TAG_{UL} (which is equal to TAG_F in this step) and the TAG sent by the SD is performed in order to authenticate the SD.

However, in order to reduce the latency due to this decryption, TAG_{UL} is ciphered with K_{req} in waiting the update command. Then, a comparison between the two ciphered TAGs is performed as soon as the SD one is received. With this improvement, steps 1, 2 and 3 (respectively Power-up, First power-up and Authentication) are performed before receiving the SD's update command.

Moreover, this is also important in terms of resources if the 3-DES block cipher is replaced by an AES one. Contrary to 3-DES engine, AES requires a non-negligible extra-cost in resources to provide decipher mode when cipher is already implemented (or cipher mode when decipher is already implemented). Thanks to the previous trick, only cipher operation is necessary and this extra-cost is not effective because it allows not using a decipher in addition to the cipher.

2.2.5.2 Results

The table 2.1 summarizes the overhead in terms of clock cycles and time for each step. The design is clocking at 60 MHz. For each step the total overhead is calculated.

To evaluate the performance overhead of this protocol implementation, only step 1 (Power-up) should be considered.

- On each power-up: Step 1 (Power-up) is performed before starting the user design in order to verify the TAG version.
- On each update command: Step 4 (TAG_F incrementing) is performed before receiving a new bitstream version. For the acknowledgment sending, network latency should be also considered, and depends only on the network type.

Steps 2 and 3 (respectively First power-up and Authentication) are performed during the user design execution and do not increase the mechanism performance overhead.

TAG_F: Flash memory tag
TAG_{UL}: User logic tag
Ek(M): Encryption of M with K as cipher key
CTAGKx: Tag ciphered by K_x

Step 1: Power-up

```
1   Read (TAG_F)
2   if (TAG_F ≠ TAG_UL) then
3     goto 22
4   end if;
```

Step 2: First power-up

```
5   Read (flag)
6   if (flag = true) then
7     Read (K_ack2)
8     CTAGKack2 := E_Kack2(TAG_UL)
9     Send(CTAGKack2)
10  end if;
```

Step 3: Authentication

```
11  Read (K_req)
12  CTAGKreq := E_Kreq(TAG_UL)
13  Read (K_ack1)
14  CTAGKack1 := E_Kack1(TAG_UL)
15  Wait for CMD
16  If (CMD = CTAGKreq) then
```

Step 4: TAG_F incrementing

```
17    Write (TAG_F+1)
18    Send(CTAGKack1)
19  Else
20    goto 15
21  end if;
22  SYSTEM SHUTDOWN
```

Figure 2.5: Security protocol implementation on FPGA

Step 4 is not considered because performance overhead is not significant face to FPGA programming (several seconds). Considering all these elements, the performance overhead is estimated at only 54 cycles: step 1 (Power-up).

The table 2.2 summarizes the overhead in terms of area. It shows the FPGA part occupied by each component of this secure remote update mechanism implementation.

Step	# Cycles	Duration (ns) F=60MHz
1. Power-up	54	900
Read TAG_F + Read flag	54	900
2. First power-up	187	3,117
Read K_{ack2}	47	783
Write Flag + EK_{ack2}(Tag)	140	2,333
Send CTAGKack2	N / A	N / A
3. Authentication	175	2,917
Read K_{req}	79	1,317
Read K_{ack1} + Ek_{req}(Tag)	48	800
EK_{ack1}(Tag)	48	800
4. TAGF incrementing	108	1,800
Write TAG_F + Write flag	108	1,800
Send CTAGKack1	N / A	N / A
TOTAL	***524***	***8,733***

Table 2.1: Performance overhead for the AFS600 device

Entity	# Tiles	Fraction of AFS600
3-DES	1,305	9%
RS232	418	3%
Flash Controller	1,005	7%
Master FSM	777	6%
TOTAL	***3,505***	***25%***

Table 2.2: Area overhead for the AFS600 device

This area overhead can be relativized considering that 3-DES, RS232 and flash memory controller can be reused by the SD. Only Master FSM cannot be reused.

Flash memory cost is not shown because it is not significant: 672 bits (including 32 bits for the first power-up flag) on the 4 Mbits (0.016%). The SD can use the rest of this physical flash memory for its own purposes. It is also possible, but not very gainful, to use only 63 bits for TAG_F. In this case, bit 64 is the flag in order to use only 640 bits instead of 672. Bit 64 of the cipher input data may be zero.

2.2.6 Discussion

Remote update of FPGA-based systems is a challenging issue from a security point of view. Section 2.2.1 showed that existing mechanisms to ensure bitstream confidentiality and integrity via encryption and authentication fail to prevent bitstream replay and thus system downgrade.

This work proposed a new communication protocol between the SD and an FPGA platform to update the FPGA configuration while preserving its confidentiality and integrity. This protocol also provides a protection against replay attacks and detects update failures.Moreover, the corresponding area overhead, when considering core reusability, and performance overhead are negligible.

This section highlights the need to embedded flash memory into the FPGA chip for security solutions in order to prevent bitstream.

In the future, K_{req}, K_{ack1} and K_{ack2} could be generated thanks to a PUF (Physically Unclonable Function) [Verayo]. A PUF is a physical structure that provides randomness. This randomness is introduced during the manufacturing process and cannot be controlled. It can be a good solution to ensure a secure unique number generation in order to identify a FPGA.

2.3 SRAM FPGAs

SRAM FPGAs are increasingly used in embedded systems for many applications. Their ability to be deployed on-field and remotely updated, makes them essential when HoD (Hardware on Demand) is required. The mobility of these devices requires efficient, flexible and adaptive approaches while having low-power consumption. In this context, DPR stands out as being a significant advantage. DPR makes run-time reconfiguration possible to allow reducing chip area [Eldredge 1996] and power consumption [Lie 2009].

Although the ability to be remotely updated can fix design vulnerabilities, it also exposes FPGA-based systems to security attacks, related to hostile environments in which they could operate. For instance, replay attacks allow an attacker to downgrade a system in order to re-expose it to outdated vulnerabilities.

Nowadays, DPR is only provided by SRAM FPGAs which are harder to secure because of their volatile nature: there is no possibility to store version references on chip. In addition to provide common security services such as confidentiality, integrity, and authenticity of bitstream, it is necessary to ensure bitstreams up-to-dateness.

Currently, no solution for DPR addresses replay attacks. This is for these reasons that it is important to protect such systems embedding FPGAs and that is why the main concern of this paper is to propose a new approach to secure updates against replay attacks addressing DPR.

2.3.1 Problem Formulation

The terminology used here is the one stated in Section 1.3.3. RP and RM are noted RP_x and RM_{xy}: x being the identifier of the RP and y being the identifier of the

RM.

2.3.1.1 Threat Models

The threat model does not consider side channel attacks, introduced by Kocher et al. in [Kocher 1996], destruction and power off DoSs. This work especially focuses on bitstream spoofing and replay attacks. In a spoofing attack, the adversary simply replaces a data by one of his own.

Replay attacks are particularly dangerous because the current approaches proposed by FPGA vendors [Xilinx 2010c] to ensure bitstream confidentiality and integrity are inefficient against replay. Even if an update may typically be performed to correct a critical security flaw, it is possible to downgrade a system, in order to exploit vulnerabilities present in a previous version.

This work proposes two case studies, each of them offering different trade-offs and described in the following sections. They match the two different threat models as described into Figures 2.6 and 2.7.

Case 1: BiT (Board is Trusted)

As shown in Figure 2.6, the first case, named BiT (Board is Trusted), considers that the adversary is unable to have physical access to the device and therefore he cannot proceed to passive or active attacks. The only way to tamper the system is through the network. Of course man-in-the-middle attacks are considered.

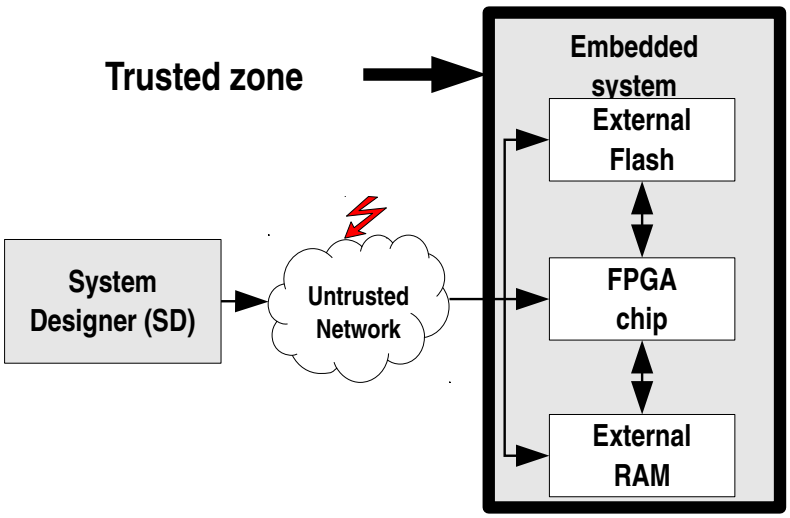


Figure 2.6: In BiT case, the whole board is trusted

Case 2: FPGA is Trusted (FiT)

As shown in Figure 2.7, the second case, named FiT (FPGA is Trusted), considers the adversary to have physical access to the device. Attackers are able to proceed to passive and active (injection) probing attacks. All attacks that allow reading, modifying or replaying the bitstream directly on the board or through the network are considered.

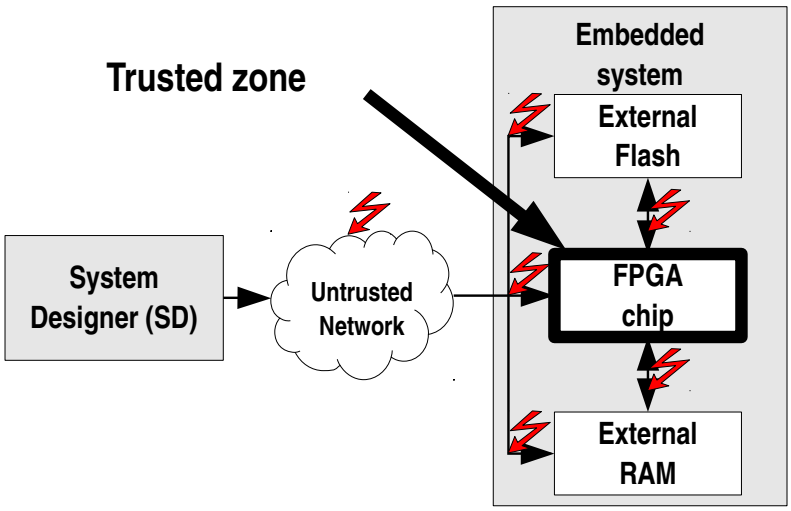


Figure 2.7: In FiT case, only the FPGA chip is trusted

In this case we need to secure remote update (the SD sends a new bitstream to the flash) and partial reconfiguration (the FPGA loads the bitstream stored in the flash memory).

2.3.2 Related works

Bossuet et al. [Bossuet 2004] use DPR and self-reconfiguration in order to secure the bitstream of SRAM FPGAs. In this scheme, the secret key is stored in the UL static area. By this way, it does not need non-volatile memories or external batteries to store the secret key. It does not secure the DPR but it uses it and self-reconfiguration capabilities to secure the bitstream of SRAM FPGAs. This solution allows the SD to override the problems related to the CL that cannot be configured by the SD.

Zeineddini et al. [Zeineddini 2005] protect partial bitstreams with encryption and authentication. AES is used for encryption and HMAC-SHA1 (HMAC (Hash-based MAC)-SHA (Secure Hash Algorithm)1) for authentication. This mechanism has been implemented on a Xilinx ML310 board. Two schemes have been compared in terms of area and performances: one based on a hard-wired PowerPC and the

other on a MicroBlaze soft processor. Their threat model consider the board as a trusted area: the partial bitstreams are stored in plaintext in external memory.

Hori et al. [Hori 2008] similarly to Zeineddini et al., propose to secure partial bitstream with confidentiality and authenticity. However, they use AES-GCM (AES in GCM (Galois Counter Mode) mode) on a Virtex-5 XC5VLX50T FPGA and reduce the trusted area to the FPGA chip. In this approach the partial bitstreams are not stored. They are sent through the UART (Universal Asynchronous Receiver Transmitter) to be used directly for reconfiguration. The system cannot process to a self-reconfiguration without requests and waits for a partial bitstream. This solution is difficult to carry out for an industrial context.

Kepa et al. [Kepa 2010] propose the SecReCon architecture. This method, based on a RoT (Root of Trust), requires involving a trusted authority and is far too heavy and complicated to implement.

None of these works preclude replay attacks.

Drimer et al. [Drimer 2009] address replay attacks with a threat model corresponding to our BiT case. However, this work is not suitable for DPR.

Similarly, we have presented in [Badrignans 2008] and [Devic 2010] two protocols that ensure up-to-dateness but with a threat model matching our FiT case for non-volatile FPGAs embedding a non-volatile memory. But once again these works does not deal with DPR.

The problematic of securing DPR has been already addressed. However, no current solution for DPR deals with replay attacks. This paper attempts to fill this gap by proposing a robust solution.

2.3.3 SecURe DPR Design Architecture

The SecURe DPR protocol aims to secure hardware updates against tampering and more particularly to preclude replay attacks. In order to be industrially convenient, partial bitstreams are stored on-board in order to process to self-reconfiguration at any time. We consider partial bitstreams to be generated using the module-based flow. Figure 2.8 presents the protocol to update a full or partial bitstream with the SecURe DPR protocol. The next two parts describe this protocol more precisely highlighting the differences between the BiT and FiT cases.

2.3.3.1 SecURe DPR: the BiT case

We remain that in BiT case, the whole board is trusted. Figure 2.9 shows the SecURe DPR architecture for this case.

In order to update the system, the SD sends a ciphered frame containing a full bitstream or a partial bitstream to the FPGA. This frame is processed and stored in external RAM memory (step 1 in Figure 2.9). This process consists in deciphering

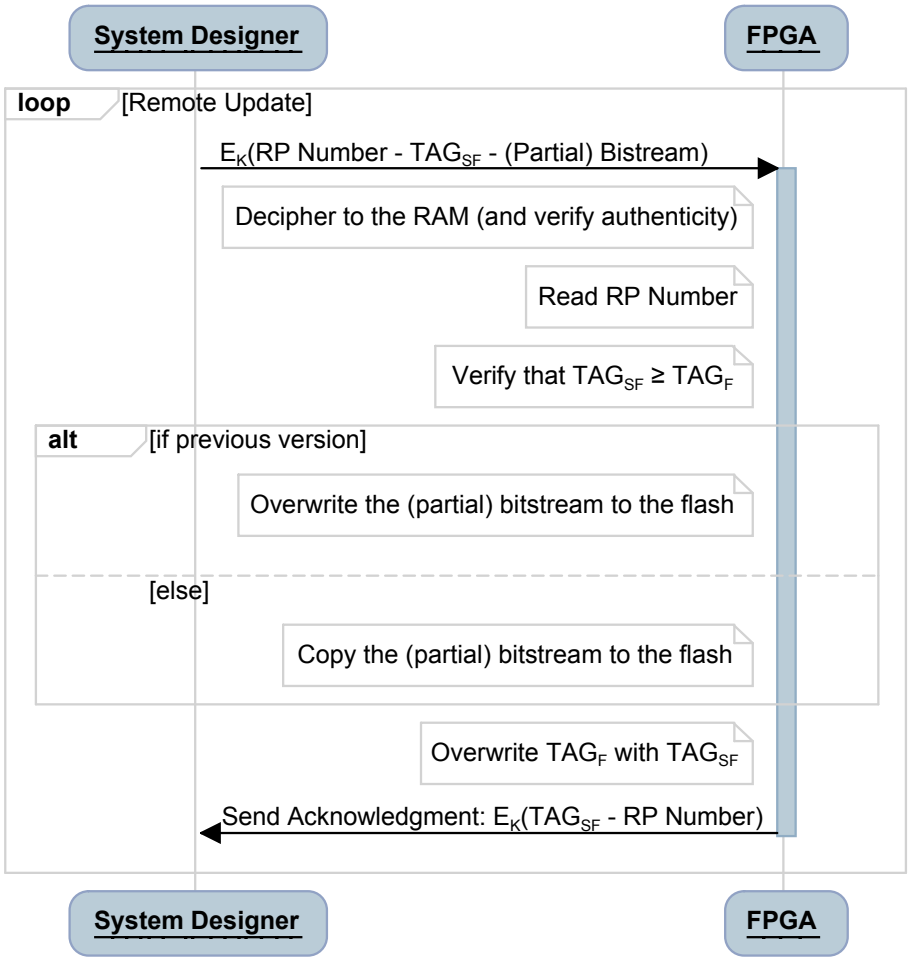


Figure 2.8: SecURe DPR protocol diagram

and verifying the integrity and authenticity of the frame. Confidentiality is ensured by AES symmetric block cipher, while integrity and authenticity are verified with a HMAC algorithm. The keys are stored in the UL static area.

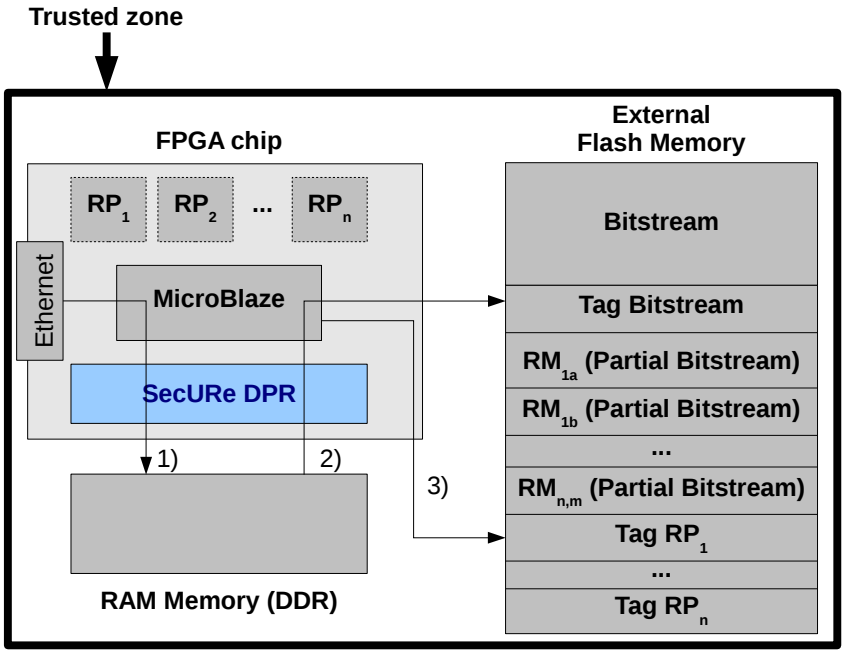


Figure 2.9: SecURe DPR architecture for the BiT case

Figure 2.10 shows the frame before and after being deciphered. As shown in this figure, after being deciphered, the frame contains three parts:

- The RP number: the identification number of the reconfigurable partition.
- The Tag: the version of reconfigurable module(s) corresponding to the reconfigurable partition identified by the RP number.
- The full bitstream or the partial bitstream: a whole bitstream is distinguished from a partial one by its RP number that is equal to zero.

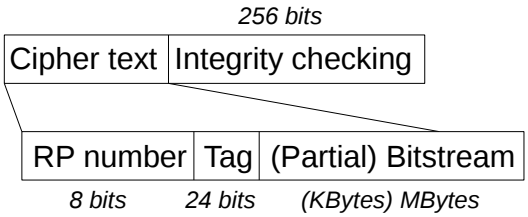


Figure 2.10: Composition of a frame before and after being deciphered

The system then reads the corresponding Tag stored in flash memory (TAG_F) and compares it with the Tag of the sent frame (TAG_{SF}). TAG_F is evaluated. If

TAG_{SF} is lower than TAG_F, then replay attack has been attempted: a signal in the design acting as an alarm is triggered to the system to react to the attack according to the system security policies. If TAG_{SF} is equal or higher than TAG_F, the full or partial bitstream must be stored in the file system of the external flash memory (step 2 in Figure 2.9). There are two possibilities:

- Update - If a full or partial bitstream with the same name already exists, the system overwrites it in the flash memory.
- Extension - If this is the first full or partial bitstream with this name, it is stored in the flash memory.

Then, TAG_F is overwritten by TAG_{SF} in the external flash memory (step 3 in Figure 2.9).

Optionally, an acknowledgment (TAG_{SF} and RP number zero-padded and encrypted) can be sent back to the SD to inform him that the update is well done.

2.3.3.2 SecURe DPR: the FiT case

We remain that in FiT case, only the FPGA chip is trusted. Figure 2.11 shows the SecURe DPR architecture for this case.

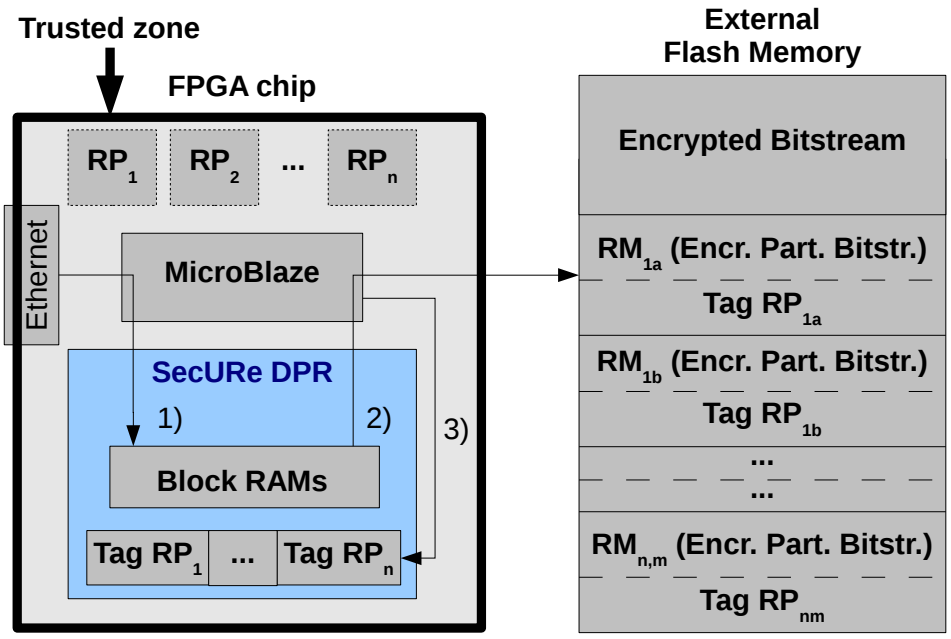


Figure 2.11: SecURe DPR architecture for the FiT case

In this case, BRAMs (Block RAMs) replace the part of the external RAM memory used to temporally store the received partial bitstream. Similarly to the

BiT case, the SD sends a ciphered frame containing a partial bitstream to the FPGA to update the system. The frame is formed with the same elements as in FiT case (refer to Figure 2.10). The difference is that the frame is not deciphered. Integrity and authenticity are verified with HMAC (step 1 in Figure 2.11). Only the beginning of the frame is deciphered to know the Tag and the RP Number. Like in the previous, TAG_{SF} is verified. It is not compared to TAG_F but to TAG_{UL}, the Tag contained in the UL. Everything then goes as described previously in BiT case except for the whole frame containing the partial bitstream which is stored ciphered with RP number, Tag and integrity verification in the flash memory (step 2 in Figure 2.11). Then, TAG_F is overwritten by TAG_{SF} in the static area of UL (step 3 in Figure 2.11).

In this scheme, the security is higher but each partial reconfiguration requires deciphering and verifying the integrity and authenticity of the partial bitstream.

The designer should choose the best option corresponding to the required security and performances.

2.3.3.3 Security analysis

With the SecURe DPR protocol, the cipher must be in a chaining mode and the Tag must be ciphered with the bitstream. HMAC uses the ciphertext as input message. This avoids deciphering the whole frame to perform integrity and authenticity checking in FiT case without compromising robustness. It poses no lack of security since HMAC is used instead of non-keyed hash functions which perform cryptographic operations on plaintext before encryption.

In FiT case, probing is considered but full bitstream is not protected against replay attacks. Therefore, the full bitstream should not be updated or the replay should not be a critical issue in this case. However, the FPGA vendor's mechanism preserves authenticity and confidentiality of the bitstream containing the AES and HMAC keys (stored in UL static area). Also in FiT case, we consider that the system is never switched-off. Actually, when the system turned-off, every current versions of partial bitstreams are lost and after system power-on, the FPGA must request the current Tags to a server. To preclude replaying this Tag transfer, the FPGA must send a Nonce (Number used once), ciphered with the key contained in the full bitstream, to the server. The latter replies by sending Tags and a Nonce derived from the received Nonce.

There are situations where both BiT and FiT cases are not sufficient in terms of security: the adversary is able to perform passive and active bus probing and the up-to-dateness of the full bitstream is critical. In these situations, it is possible to limit such type of physical access by using specific IC packages. Approaches [MacPherson 1994] offers a detection mechanism against enclosure

tampering. This enclosure wraps several components and their interconnections. In case of tampering, it generates an alarm triggered by package sensors.

A solution could be that the FPGA vendors embed a non-volatile memory, even small-sized, in the chip in order to have the benefits of proposed solutions for both cases BiT and FiT. In this case the Tags would be stored in the embedded non-volatile memory. By this way problems related to the volatility of BRAMs would be avoided. Moreover, as partial bitstreams are small, they could be stored directly in plaintext in the embedded non-volatile memory and it would no longer be necessary to consider two cases.

2.3.4 Results

The implementation of our SecURe DPR protocol is based on a ML605 Xilinx development board embedding a Virtex-6 XC6VLX240T. The AES-CBC (AES in CBC (Cipher Block Chaining) mode) core is iterative and provides both encryption and decryption. It computes 128-bits in 11 clock cycles. HMAC algorithm is based on folded SHA-256. Each 256-bits computation takes 64 clock cycles. Data are sent by the processor through the AXI (Advanced eXtensible Interface) AMBA (Advanced Microcontroller Bus Architecture) bus.

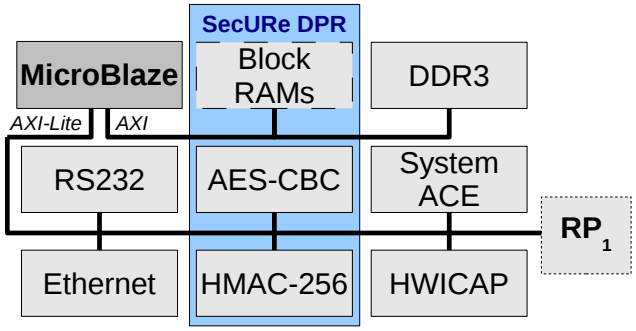


Figure 2.12: Secure DPR implementation overview

This design presented in Figure 2.12 was realized with the Xilinx 13.4 design flow (XPS (Xilinx Platform Studio), SDK (Software Development Kit), and PlanAhead). The processor is a MicroBlaze soft core operating at 100 Mhz with 8 KB of instruction cache and 8 KB of data cache. Xilfatfs has been used to access the file system on the CF (Compact Flash) memory. The size of partial bitstreams is 23 KB.

All results have been provided post PAR (Place And Route) process.

	Write to Flash		SecURe DPR overhead		
Case	**# Cycles**	**Duration (ms)**	**# Cycles**	**Duration (ms)**	**Ratio**
BiT case	15,202,273	15	8,319,467	8	**54.73%**
FiT case	13,643,969	14	3,864,052	4	**28.32%**

(a) Update performance overhead

	Reconfiguration		SecURe DPR overhead		
Case	**# Cycles**	**Duration (ms)**	**# Cycles**	**Duration (ms)**	**Ratio**
BiT case	58,248,054	58	0	0	**0.00%**
FiT case	58,116,629	58	8,182,029	8	**14.08%**

(b) Reconfiguration performance overhead

Table 2.3: Performance overhead

2.3.4.1 Performance evaluation

Tables 2.3(a) and 2.3(b) provide a summary of performances overhead for the two case studies described in Section 2.3.1.1 (BiT case) and Section 2.3.1.1 (FiT case). Table 2.3(a) shows the performance overhead of a partial bitstream update, while Table 2.3(b) highlights the performance overhead of a partial reconfiguration. Since, reconfiguration occurs more frequently than update, SecURe DPR has been designed to impact more updates than reconfigurations.

2.3.4.2 Area evaluation

Tables 2.4(a), 2.4(b) and 2.4(c) present the area overhead.

Table 2.4(a) shows the area for the base system without security detailing each component (MicroBlaze, DDR3 (DDR (Double Data Rate) 3rd generation) controller, Ethernet controller, AXI and AXI lite buses, HWICAP (HardWare ICAP), RS232 UART and System ACE (System Advanced Configuration Environment) CF wrapper). Tables 2.4(b) and 2.4(c) highlight the area overhead for the two use cases (BiT and FiT) corresponding to the two threat models described in Section 2.3.1.1. The only difference for the FiT case architecture is the need of 16 36Kb-BRAMs to avoid using external RAM. These BRAMs allow processing up to 64 KB-sized partial bitstreams. The number of BRAMs can be modified to fit the system requirements. This area overhead is compared to a lightened base system embedding an area-optimized MicroBlaze (3-stage pipelined and implemented with DSPs). In addition, it can be relativized considering AES-CBC and HMAC-SHA256

Components	Slice FF	Slice LUT	BRAM	DSP48E1
MicroBlaze + Caches	2,180	2,239	8	3
DDR3 controller	3,264	3,750		
Ethernet controller	607	712	2	
AXI	624	543		
AXI-Lite	176	408		
HWICAP	477	502	1	
RS232 controller	89	116		
System ace (CF)	88	84		
Total	***7,505***	***8,354***	***11***	***3***

(a) Area for the base system

Components	Slice FF	Slice LUT	BRAM	DSP48E1
AES (+wrapper)	1,105	3,184	7	
HMAC-SHA256 (+wrapper)	1,504	1,908	1	
Total	***2,609***	***5,092***	***8***	***0***

(b) Area overhead for BiT case

Components	Slice FF	Slice LUT	BRAM	DSP48E1
AES (+wrapper)	1,105	3,184	7	
HMAC-SHA256 (+wrapper)	1,504	1,908	1	
Block RAMs (64 KB)	10	24	16	
Total	***2,619***	***5,116***	***24***	***0***

(c) Area overhead for FiT case

Table 2.4: Area overhead

(HMAC-SHA256) cores which can be reused for a targeted application.

This section shown that SecURe DPR is an efficient solution. The performance overhead for self-DPR is null or very low (14%) depending on the case (BiT or FiT). The area overhead is quasi-null if reusability is considered.

This implementation is a proof of concept. The goal is that FPGA vendors, like Xilinx, integrate the SecURe DPR protocol in the CL. Since existing mechanism [Xilinx 2010c] already provides AES and HMAC cores, the area overhead could be quasi-null: it requires mainly to modify interconnections.

2.3.5 Discussion

This section proposes the SecURe DPR protocol, to fill-in the lack in DPR context concerning replay attacks. To the best of our knowledge, it is the first mechanism presenting a solution against replay attacks for DPR. SecURe DPR provides a protection ensuring partial bitstream up-to-dateness while preserving confidentiality, integrity and authenticity. This work suggests solutions to the threat models corresponding to industrially-driven use cases of self-reconfiguration. An implementation has been done to demonstrate the feasibility, and evaluate the overhead in terms of performance and area. The results show that SecURe DPR is a very efficient solution that can be used as it stands or integrated in the CL by FPGA vendors.

This section highlights that the addition of a non-volatile memory embedded in the FPGA chip would increase the security of embedded systems based on FPGAs.

2.4 Conclusion

Remote update of FPGA-based embedded systems is a challenging issue from a security standpoint. We showed that existing mechanisms aiming to provide bitstream confidentiality and integrity with encryption and authentication fail to preclude bitstream replays and thus system downgrades. It is surprising that the FPGA vendors do not protect their devices against replay attacks which are serious threats, easy to perform for an attacker with basic equipments.

This chapter proposes very efficient solutions for both Flash and SRAM FPGAs taking into account their strengths and weaknesses addressing several industrially-driven threat models. It covers a very large domain application by considering advanced FPGA capabilities like DPR. These bitstream securing solutions have been implemented to prove the feasibility as well as to evaluate their area and performance overheads.

This work is a sound basis for securing embedded systems based on FPGA technologies. Indeed, now that we can trust the FPGA configuration, it is possible

to continue our efforts to secure the software which will operate on the FPGA. That is the subject of the next chapter.

Chapter 3

OS and Application Security

- What can you see, Neo?
- It's strange... the code is somehow different.
- Encrypted?

Morpheus and Neo - Matrix Reloaded

Contents

This chapter describes how to secure OSs and applications running on previously trusted FPGAs. It presents an overview of currently used counter-measures to protect OSs and applications on FPGAs. It summarizes, organizes and explains existing counter-measures which are not necessarily specific to FPGAs. It also proposes an OS boot verification that allows precluding malicious kernel modifications. This work also ensures kernel up-to-dateness and supports updates management. It highlights the performance improvement of this security mechanism thanks to hardware acceleration mechanisms. It also describes the implementation and evaluates the overhead of this solution in terms of performance and area.

3.1 Introduction

Embedded systems require more and more security. Due to the mobility feature, these systems could be exposed to an hostile environment.

The constant complexity growing of this kind of devices, that can be FPGAs, involves to embed an OS.

An attacker can inject, through memory or another way, malicious code during the OS execution, taking for instance entire control of the system.

[Huang 2003] is a good example of attack. In this paper, Huang explains his attack on the X-Box (the Microsoft game console), designed with a classical computer architecture. This game console has been hacked by analyzing and modifying the processor data and instructions during the execution.

AEGIS [Suh 2003] is an architecture demonstrating a complete OS securing from hardware to software. However, it requires to have a dedicated processor with a modified OS.

Data theft or malicious code injection can be included into a generalist model addressing access rights. We remind that considered threat covers spoofing and replay.

This chapter is composed by two parts. First, Section 3.2 describes how to protect OSs and applications on FPGAs but are not specific to FPGAs. Then, Section 3.4 describes how to protect the processor kernel boot on FPGAs.

3.2 OS and application execution security

3.2.1 Protecting memories

3.2.1.1 Protecting RAM

In order to preclude an attacker to extract or modify critical data, the RAM memory and RAM data transfers should be protected with confidentiality, integrity and up-to-dateness. This memory requires high performances and is a critical component in terms of security. Due to its wide area, RAM cannot be integrated in the processor chip. Its random access behavior makes impossible to use mechanisms similar to bitstream protection. The X-box, has been hacked because the RAM was weakly encrypted.

Merkle-trees (refer to figure 3.1) are an interesting solution. Elbaz proposes a new type of integrity trees, PRV (Protected Random Value)-trees approach [Elbaz 2006], a counter-measure based on Merkle-trees concept. PRV-trees ensure confidentiality, integrity and up-to-dateness and require only to store on chip the root of the tree allowing to verify the whole tree and therefore the entire RAM. With FPGAs, it is possible to store several roots, for instance in block RAM, in order to

improve performances.

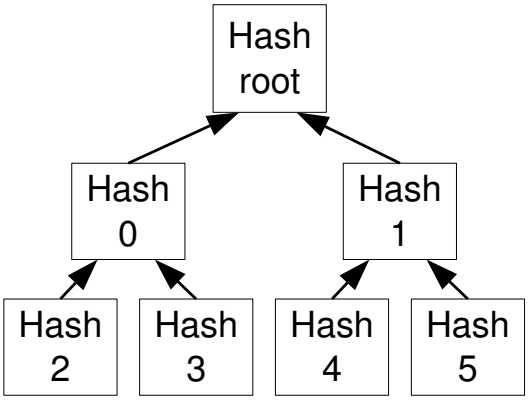


Figure 3.1: A 2-ary Merkle-Tree (or Hash-Tree)

In [Vaslin 2009], Vaslin et al. propose a high-efficiency solution based on OTP (One-Time Pad or One-Time Password) encryption and CRC. They explain more precisely the mechanisms enumerated in this section.

Crenne et al. continue Vaslin's works in [Crenne 2011] offering a more secure solution based on AES GCM instead of CRC. This solution allows having different memory security policies for different application tasks.

All current solutions to secure RAM impact seriously the system performances.

3.2.1.2 Protecting file system

The file system is a method of storing and organizing data into an easy-to-manipulate database. It should be protected against the same type of attacks as RAM. However, contrarily to RAM, the file system is stored in a NVM. That is why a non-volatile value (like a root of Merkle-trees) has to be stored in a NVM into the FPGA chip.

For a long time, tools are available to encrypt embedded Linux file systems. *eCryptfs* [Moog] provides file system level encryption. *dm-crypt* [Saout 2004], the *cryptoloop* successor, provides block device encryption.

In a classical X86 architecture, such mechanisms require to enter manually a password at each system power-up to allow the processor to decrypt the file system. With FPGAs, it is possible to store and hide the secret decryption key inside the bitstream, making the system autonomous and able to run without a human intervention.

3.2.2 Resources isolation

In complex systems, a good way to avoid intrusions is to use the concept of resources isolation. The principle is to isolate each application or component of the system. In this way, malicious applications cannot tamper other ones.

3.2.2.1 TPM: Trusted Platform Module

The TCG (Trusted Computing Group) has proposed TPM (Trusted Platform Module) specifications [TPM]. TPM is a component providing cryptographic functionalities and allowing to store keys. This component runs like a smartcard.

Figure 3.2 shows a TPM daughterboard plugged on a personal computer motherboard. In this case, it can be used to verify the platform integrity and to store passwords or keys.

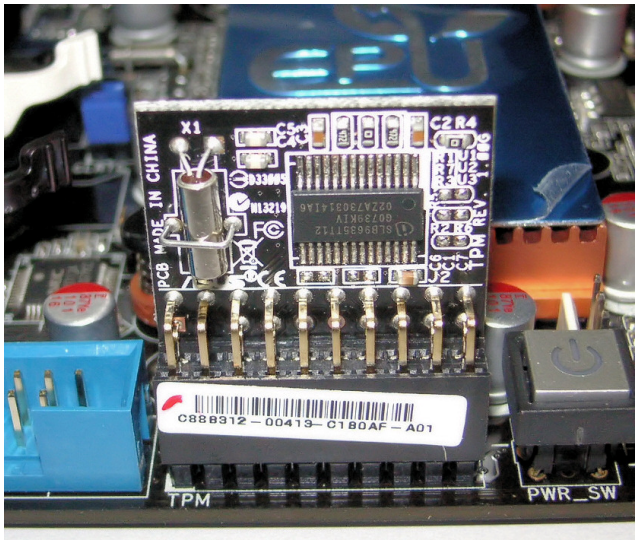


Figure 3.2: TPM security device plugged on a personal computer motherboard

[Eisenbarth 2007] describes how to embed a TPM into an FPGA chip. With FPGAs, the system designer can do everything in term of isolation (keys non-accessible from the processor, for example).

3.2.2.2 Hardware firewalls

Like applications, IPs can be coded by anyone. That is why [Cotret 2011] proposes an architecture with firewalls inserted at each interface between IP and bus. This architecture prevents an attacker to retrieve or modify passwords, keys or other critical data.

3.2.2.3 SandBoxing

Devices are more and more open in order to offer still more services to consumer. Smartphones illustrate well this trend with the possibility to pick applications, coded by anyone, from application stores. These applications can have malicious behaviors. They can be created to retrieve confidential data or simply coded using bad practices. In both of these two cases, the applications may affect others.

To protect the system against such applications, it becomes usual to confine them in sandboxes. A sandbox is a superintended environment offering a restricted set of resources. Peripheral accesses or critical parts of host system are generally disabled or restricted. For example, confining an application to a dedicated memory range and disabling the network is a good way to prevent sending data over the network. Figure 3.3 explains sandbox principle by comparing the impact of an untrusted program execution on the system memory.

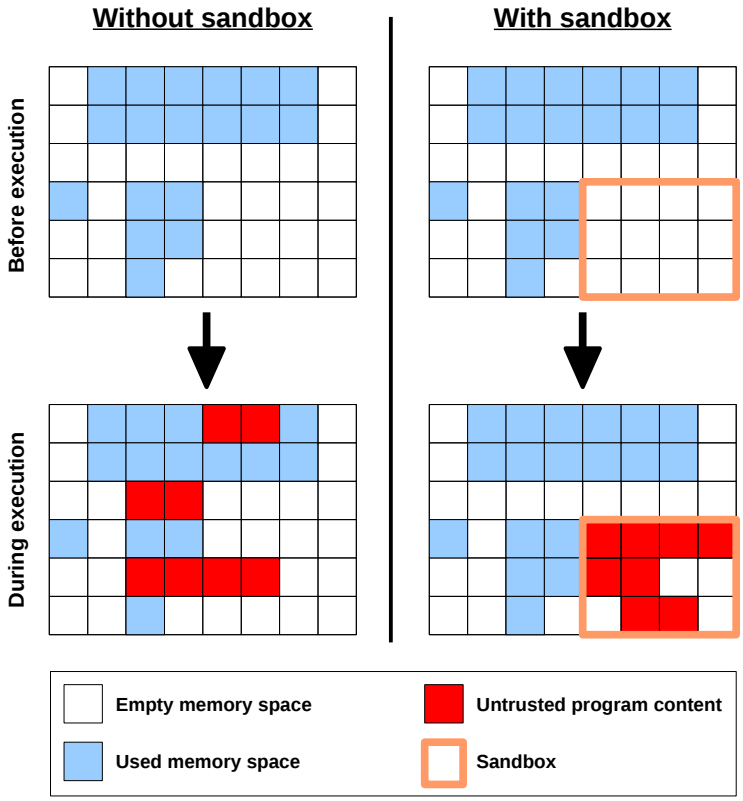


Figure 3.3: Untrusted program execution with and without sandboxing isolation

Sandbox systems are presented in [Tzur 2004] for traditional OSs and in [Bläsing 2010] for Android smartphones. It is possible to apply sandboxing on FPGAs especially since the most current soft processor architectures involve a MMU (Memory Management Unit) (refer to section 3.2.2.4).

3.2.2.4 Virtualization Approach

Virtualization is an approach to isolate resources. A virtualized component is not aware about its real environment, and by this way, it cannot reach directly another component.

Actual computer architectures offer hardware supports for processors, memory and peripherals virtualization.

VT-x (Virtualization Technology) [Uhlig 2005] for Intel and AMD-V (AMD (Advanced Micro Devices) virtualization) [AMD 2005] for AMD offer a hardware support for processors virtualization. This adds a new execution mode, more privileged than kernel mode. It helps hypervisor to easily catch critical instructions from VMs (Virtual Machines).

MMU allows protecting memory against code execution. It prevents malicious programs from accessing memory which it does not have access rights. It avoids may buffer overflows or dumping attacks.

Intel and AMD with respectively VT-d (Virtualization Technology for Directed I/O) [Intel 2011] and IOMMU (Input / Output Memory Management Unit) [AMD 2009] protect memory against peripherals like MMU protects memory against processor. It prevents malicious devices from accessing memory which it does not have access rights. It also avoids DMA (Direct Memory Access) attacks. DMA allows a peripheral to perform memory transfers without involving the processor. Figure 3.4 shows how MMU and IOMMU protect memory by isolation technique.

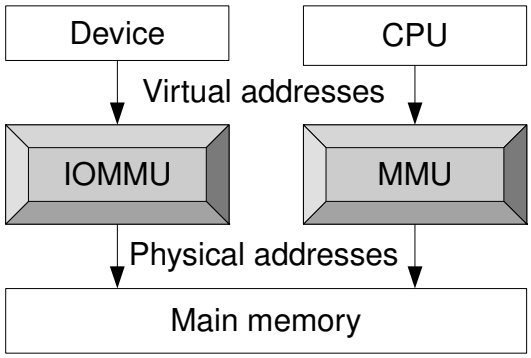


Figure 3.4: MMU and IOMMU protecting memory by isolation

[Sang 2010] describes a DMA attack performed on a VT-d and a non-VT-d protected computers. It demonstrates the efficiency of an Intel VT-d by DMA attacking a Linux machine. This attack by code injection through the FireWire port allows performing *privileges escalation*. This attack succeeds when VT-d technology is disabled and failed with an unauthorized access message error from DMAR (Direct Memory Access Remapping) when it is enabled.

[Sang 2010] also highlights security vulnerabilities of this counter-measure. It explains that a malicious peripheral can access unauthorized memory range by pretending to be another one.

Qubes OS [Rutkowska 2010] is based on a *Xen* hypervisor. In this scheme, software isolation is ensured by the hypervisor. Security is based on VMs impermeability. In fact, two processes in two different VMs can be much better isolated between each other than if they are together in the same monolithic kernel. In FPGA context, classical virtualization is possible but has too much impact in

terms of performance overhead.

In another approach, TrustZone [ARM 2009] by ARM (Advanced RISC (Reduced Instruction Set Computing) Machines), isolates two modes by virtualization: a secure and an insecure one. Here a *secure bit* is added to the initial bus in the hardware. OS execution is protected and monitored thanks to a hypervisor enabling the secure bit when is switched to secure mode. TrustZone can be implemented on FPGAs because of its lightness. Indeed, the involving hypervisor is very simple and the IOMMU is reduced to adding the secure bit mechanism. Figure 3.5 is an overview highlighting isolation by virtualization in TrustZone architecture.

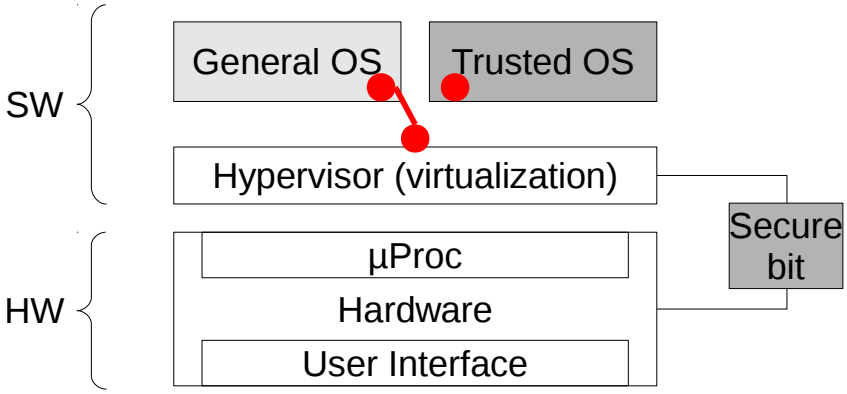


Figure 3.5: ARM TrustZone approach

Currently, Xilinx offers AXI bus implementation supporting the TrustZone secure bit (refer to [Xilinx 2010a]).

In this section, we highlight that isolation was ensured by hypervisor's supervision. Obviously, it considerably reduces the attack surface but security flaws and attacks were shifted to hypervisors. In [IBM 2010], IBM offers a detailed study on virtualization vulnerabilities and highlights that isolation can be affected by *escape to host* or *escape to hypervisor* flaws. More precisely, XBOX360, the X-BOX successor, was better secured thanks to virtualization but was broken exploiting a hypervisor vulnerability (refer to [Keurentjes 2007]).

3.3 Discussion

This survey is an overview of currently used counter-measures to protect FPGA-based devices embedding an OS and their vulnerabilities. Despite the fact that reconfigurable architectures imply the use of specific counter-measures for FPGA-specific problematics, some traditional computing mechanisms can be reused or nevertheless adapted to FPGAs.

It highlights that RAM protections against hardware probing or injections are very penalizing in terms of performances.

TrustZone on FPGAs should be more investigated in order to use FPGA flexibility to offer innovative solutions.

The aim in such complex designs is to reduce the attack surface: over-intricate counter-measures are more flaw-prone.

3.4 OS boot security

Protecting the OS execution is a good thing. However, it is useless if the kernel boot is not trusted.

To protect OS boot on FPGA it is necessary to trust progressively the different parts during the system start-up, establishing a trusted bitstream-to-kernel-boot chain. That is why this work proposes a trusted computing mechanism taking into account the whole security chain from bitstream-to-kernel-boot ensuring, both hardware and software, integrity while preventing replay attacks.

For these reasons this work proposes to protect the processor kernel boot on FPGA. This kernel is typically stored in an external NVM due to the large storage capacity requirement. Generally this memory is off-chip, allowing an attacker to modify the kernel in order to introduce malicious code.

Moreover, remote update turns out to be an essential service. That is why this chapter considers that FPGAs embedding a processor are able to process OS updates through, for instance, an insecure network. This may give rise to security flaws affecting the system integrity or freshness. Integrity can be altered by spoofing or modifying data in order to introduce malicious code.

The work presented here precludes kernel modifications, prevents against replay attacks and supports updates. It highlights the performance improvement of this security mechanism thanks to an FPGA capability: the hardware acceleration.

3.4.1 Threat model

Figure 3.6 shows that the FPGA system is exposed to hostile environment where physical but non-invasive attacks, excepted side channel ones, are feasible. FPGA chip is considered as a trusted zone. Typically, off-chip bus probing and active probing (injection) are considered in our threat model.

All attacks that allow reading, modifying or replaying the kernel directly on the external memory are considered. Typically in "man-in-the-middle" attacks, an attacker is assumed to be placed on the link between the FPGA chip and the external memory where the kernel is stored. He is able to retrieve and modify all the communications transmitted through this link.

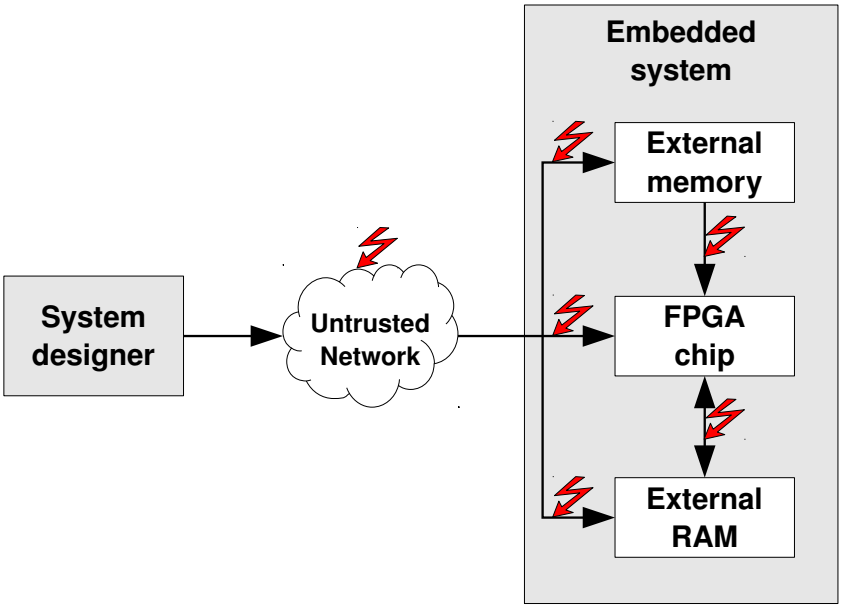


Figure 3.6: Threat model

Considering such an attacker, we focus on two main attacks:

- Spoofing: the adversary replaces a data or a partial data by his own.
- Replay: the adversary records a data and *replays* it at any time.

This threat model considers both FPGAs with and without on-chip user flash memory. It is essential that both of them embed a hard-wired mechanism protecting the bitstream confidentiality and integrity like [Microsemi 2010b] for Actel or [Xilinx 2010c] for Xilinx.

- FPGAs with on-chip user flash:

  Existing FPGAs embedding flash, like [Microsemi 2010a], [Xilinx 2009], [Lattice corporation 2010a] and [Lattice corporation 2010b], can preclude downgrades applying work [Devic 2010] explained in Chapter 2.

- FPGAs without on-chip user flash:

  Existing FPGAs cannot preclude bitstream replays. Several mechanisms against such attacks on FPGAs without on-chip user flash have been presented in [Badrignans 2010]. These mechanisms involve modifying the configuration logic, and are not currently proposed by FPGA vendors.

Considering the beginning of this section, securing the bitstream is supposed to be well done and already treated in [Devic 2010] and [Badrignans 2010]: we can assume the bitstream already secured.

In this case study, a volatile FPGA without user flash is used, focusing on the kernel boot protection and not considering bitstream downgrades assuming the existence of mechanisms like [Devic 2010] and [Badrignans 2010]. As prototyping platform, we used a Xilinx Virtex 6 XC6VLX240T on a ML605 development board but any others FPGAs able to embed a processor can be used. This point will be discussed in section 3.4.4.

3.4.2 Related works

The aim is to prevent physical and logical attacks against hardware components. For instance, [Huang 2003] explains how the X-Box (the Microsoft game console), based on a classical computer architecture, has been attacked by analyzing and modifying the processor data and instructions.

In this work we will see how to ensure kernel integrity in FPGA thanks to hash algorithms. Such algorithms are classically used to ensure kernel integrity in ASICs like in [Discretix] where Discretix used such a mechanism to securely implemented processor. It is the same principle for Atmel [Atmel 2006] and several others companies. However, this paper presents how to adapt this mechanism to reconfigurable architectures, in a first time by overriding the reconfigurable weaknesses and in a second time by using the FPGAs capabilities.

AEGIS [Suh 2003] is a well accomplished example of hardware to OS securing but requires having a specific processor with a modified operating system.

More recently, ARM conceived TrustZone [ARM 2009] to build a boot-to-OS chain of trust. In this scheme the software is protected and monitored thanks to secure bit added to the bus in the hardware. TrustZone uses RSA-PSS (Rivest, Shamir and Adleman - Probabilistic Signature Scheme) to secure the boot. This asymmetric cryptographic protocol verifies the signature of a second level bootloader but is vulnerable to replay attacks.

In section 3.4.3.3, we will use asymmetric cryptography in order to add more flexibility. An Apple patent describes a similar but more complex principle in [de Cesare 2009]. In this patent Apple uses certificates and a PKI to securely boot a code from an off-chip memory.

3.4.3 Secure boot principle

The objective of this work is to secure embedded Linux boot on FPGA preventing an attacker to:

- execute its own potentially malicious program stored in the external memory
- modify the kernel stored in the external memory
- replay an out-of-dated kernel containing known vulnerabilities

3.4.3.1 Classical Xilinx boot

First, it is important to well understand how a classical boot is working. The figure 3.7, derived from [Hill 2010], explains the standard boot process on Xilinx FPGAs.

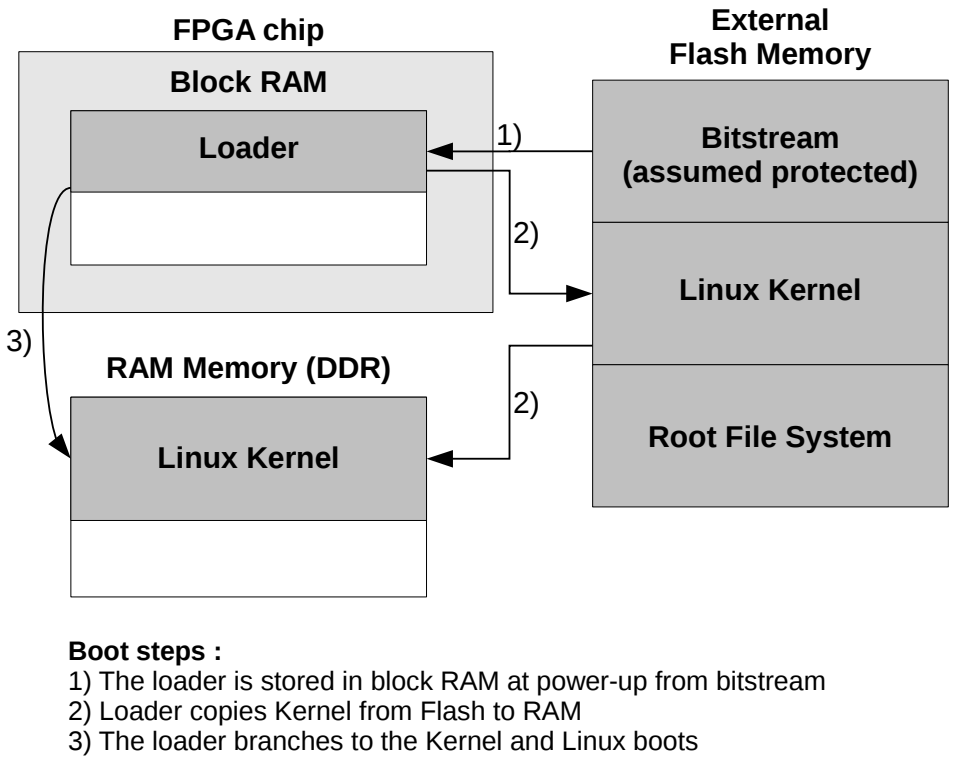


Figure 3.7: Classical Xilinx boot

First, the bitstream is copied from the external flash memory to the FPGA and started. In the same way internal FPGA BRAMs are initialized with a small-sized program called *bootloader*. It just copy the kernel from the external flash to a large-sized RAM memory. After, the bootloader jumps to the appropriate RAM address, the beginning of the kernel. Thus, the kernel is starting.

This process is unsecured because there is no integrity verification. For instance an attacker can replace the kernel by his own in order to execute his potentially malicious code.

This is an unacceptable issue when dealing with applications requiring security.

3.4.3.2 Boot with integrity verification

Hash algorithms are classically used to ensure kernel integrity in ASIC like in [Atmel 2006] where Atmel used such a mechanism to secure hardly implemented processor.

Figure 3.8 shows how to add integrity verification to a classic boot mechanism.

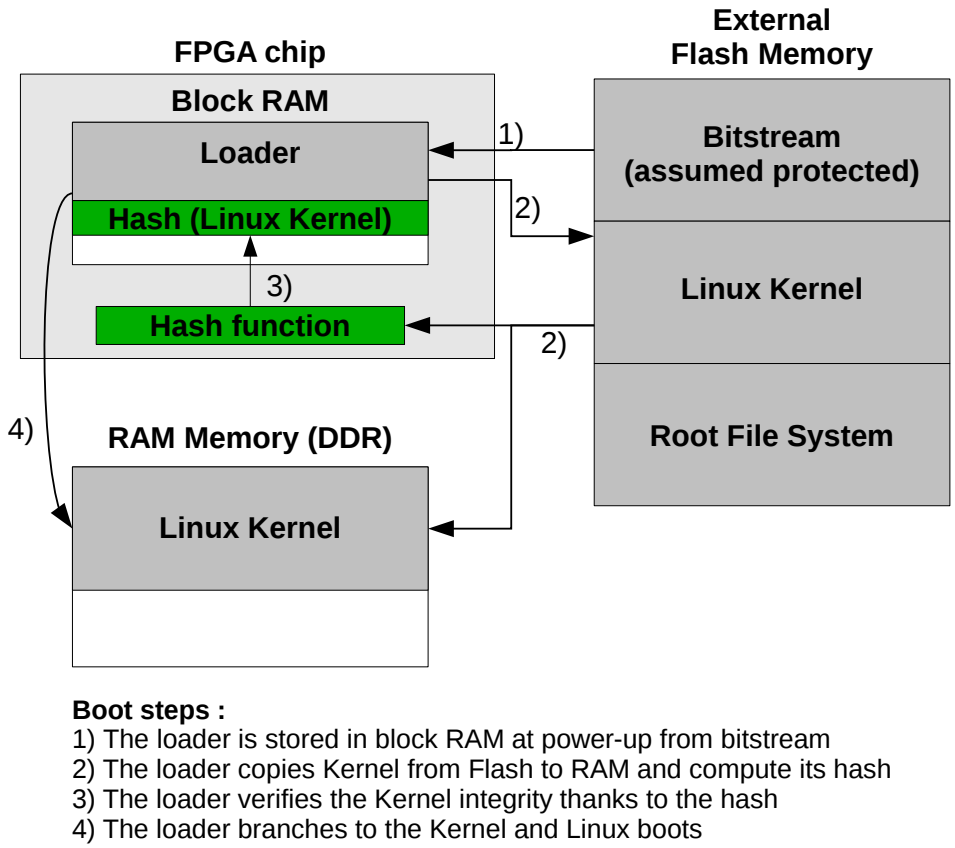


Figure 3.8: Boot with integrity verification

The bootloader is contained in BRAMs initialized by the bitstream. As explained in Chapter 2, the bitstream is assumed to be securely updated. This implies that the bootloader is trusted.

The difference with the classical boot lies in the addition of a kernel integrity verification of the copied kernel. This verification is performed by a hash algorithm implemented in the only trusted area: the FPGA chip. The kernel hash value is stored in the bootloader.

In this scheme, the kernel is copied to the RAM, as in the classical boot. But after, the hash core generates the hash value of the kernel contained in the RAM. Then, the bootloader jumps to the kernel address in the RAM only if the resulting kernel hash value is equal to the one stored in the bootloader.

In this way, a malicious program, a modified kernel or a previous kernel version cannot replace the genuine one.

This manner to do involves to change the hash value contained in the bootloader, and so the whole bitstream each time the SD updates the kernel. That is why the

section 3.4.3.3 introduces a mechanism to avoid that by adding flexibility.

3.4.3.3 Using asymmetric cryptography to add flexibility

The goal of this section is to add flexibility to the previously trusted bitstream-to-kernel-boot chain.

The flexibility improvement allows the SD to change the kernel, in external memory, without changing the bitstream. More precisely, it allows removing the performance overhead due to the bitstream downloading and FPGA programming (several seconds).

This flexible protocol is described in figure 3.9.

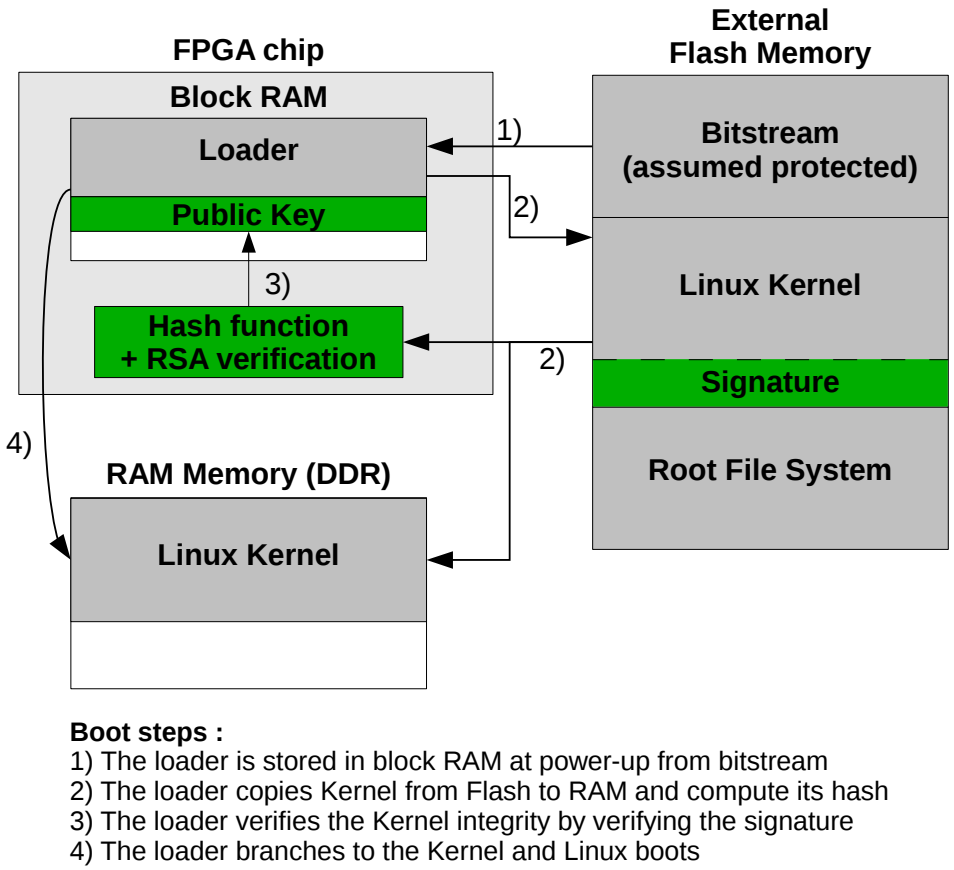


Figure 3.9: Boot with flexible integrity verification

In order to avoid storing a kernel-specific hash value in the BRAMs, asymmetric cryptography is involved.

In this scheme, the kernel provider keeps the private part of the asymmetric key pair to sign the different future kernel versions. The public key is stored in the bootloader instead of the hash value to verify the kernel signature.

Such as the previous section, a kernel hash value is processed to ensure integrity. This hash value is signed thanks to the kernel provider's private key. The generated signature is placed after the kernel in the external memory.

In this scheme, the kernel is copied to the RAM, as in the classical boot.

There are two steps to verify the kernel integrity:

- First, the Hash core generates the kernel hash value as in the boot with integrity verification.
- Then, the bootloader verifies the signature of the hash value stored in the external Flash memory with the previously generated hash value and its public key.

With this mechanism, the public key stored in the BRAMs remains the same even if the kernel is updated. Consequently, the BRAMs do not need to be reinitialized and therefore the bitstream does not require to be changed and reloaded.

In this scheme asymmetric cryptography do not increase the solution security level. The goal is only to add flexibility.

However, this flexibility may give rise to security flaws concerning replay attacks. The next section highlights this security flaws and propose several counter-measures.

In section 3.4.5, we will explained how to implement this kind of mechanisms in a reconfigurable architecture like FPGA, emphasis on area saving or performances.

3.4.4 Security analysis

This analysis focuses on replay and malicious modifications attacks. However, it is possible to add confidentiality to this mechanism by using a cipher block.

In this paper only well known algorithm, NIST approved are used:

- SHA-256 is used for integrity verification
- RSA-1024 is used for signature verification

As explained in section 3.4.3.3, the flexibility improvement allows updating the kernel, in external memory without changing the bitstream. However, this flexibility allows replay attacks. Indeed, after a kernel update without bitstream updating, an old kernel can be replayed since it has been signed with the same private key.

In order to preclude the exploitation of this security flaw, several counter-measures are proposed in the following sections:

3.4.4.1 FPGAs without on-chip user flash memory (classical SRAM FPGAs)

To secure this FPGA family, it is required to have the bitstream protected with confidentiality, integrity and against replay attacks. Currently, no SRAM

FPGA vendor provides sufficient counter-measures against replay attacks but [Badrignans 2010] proposes two solutions to solve such a problem:

- First based on regular polling.
- Second based on the modification of the configuration logic. The configuration logic is the part of the FPGA hardly implemented by the FPGA vendor.

In the context of replay-resistant SRAM FPGAs, it is possible to have two kinds of updates in order to avoid the exploitation of the security flaw introduced by the flexibility feature:

- Non-critical kernel update: in this case only the kernel is updated to benefit of the flexibility. The designer is aware that this update is vulnerable to replay attacks and can be maliciously replaced by the previous one.
- Critical kernel update: in this case both kernel and bitstream are updated. The bitstream is updated to securely change the public key. This key modification prevents from validating an older kernel signed with the previous key.

3.4.4.2 FPGAs with on-chip user flash memory

In case of such non-volatile or volatile FPGAs, it is possible to avoid striking a balance using critical / non-critical updates. The user memory allows storing securely the public key and to modify it without reinitialized BRAMs. In the same way, it is possible to store directly the hash value in this user memory. In this case, flexibility is offered by user NVM and the asymmetric cryptography is not required.

Of course, if the user flash size allows it, the kernel can be directly stored in the user flash but it usually spends too much on-chip memory.

3.4.5 Implementation and Hardware acceleration

The concepts presented in this chapter can be implemented on any FPGA able to embed a processor. This work can be applied to any OS or program.

As prototyping platform, we used a Xilinx Virtex 6 XC6VLX240T on a ML605 development board. The Flash memory used to store the Linux kernel is the 32MB BPI Flash from Numonyx available on this board.

FPGAs are particularly adapted to perform hardware acceleration. That is why, we have implemented four designs corresponding to four strategies in order to propose four different performance/area overhead trade-offs:

The first one is a minimal design to run a Xilinx embedded Linux [Xilinx] running on MicroBlaze and based on the 2.6.31 Linux kernel version. In this design,

the kernel verification is performed by the bootloader thanks to a SHA-256 function implemented in C. This implementation does not generate area overhead except increasing the bootloader size of 16 KB (equivalent to 4 36Kb-BRAMs) with -O2 compiler optimization. This design is the base system: the others implementation strategies will be compared to this one.

The second one performs hardware acceleration. It means the MicroBlaze processor does not perform the hash algorithm himself. It sends the data to an hardware IP specifically designed to performed the SHA-256 hash in order to accelerate the execution. This SHA-256 core has a folded architecture: the same logic is used for each iteration of the hash value computation. Each 256-bits computation spends 64 clock cycles. Data are sent by the processor in several packets of one 32-bits-word through the PLB (Processor Local Bus). In this case the acceleration factor is 7.7.

The third one sends the data to an hardware IP specifically designed to performed the SHA-256 hash like the second one. The only difference is the data transfer. In this design, we perform DMA: the data transfer is delegated to a XPS central DMA controller [Xilinx 2010b]. It allows transferring packets of one, eight or sixteen 32-bits-word through the PLB. In this case the acceleration factor is 70.

The last one corresponds to the flexibility improvement. This design is the third one with the addition of an RSA-1024 IP. This core computes 64 iterations of 16 bits data to use 25x18 Virtex 6 multipliers. It is only used for signature verification. That is why the bootloader contains only the public part of the key pair. This asymmetric variant can be used on the three previous designs (soft SHA-256, hard SHA-256, hard SHA-256 with DMA transfers) adding only 1 ms to the boot duration.

3.4.5.1 Performance overhead

The two tables 3.1(a) and 3.1(b) highlight the performance overhead, throughput and gain obtained through the hardware acceleration for each strategy.

These data are acquired on a ML605 development board with a Virtex 6 XC6VLX240T embedding a 100 Mhz MicroBlaze processor booting a 2.8 MB Linux kernel. Performance overheads for others kernel size can be extrapolated thanks to the throughput. In our case, taking into account the flexibility improvement, the global throughput is 65 MB/s. This throughput considers only the overhead introduced by the securing.

This performance overhead become null if the DMA controller is used to copy kernel from Flash to RAM. The classical Xilinx boot, consisting in copying kernel from Flash to RAM and branching to the appropriate address, lasts 280 ms using memcpy function or 180 ms using a DMA transfer. The corresponding bootload throughputs are respectively 10.0 MB/s and 15.6 MB/s. The whole bootload

IP	# Cycles	Boot time Overhead	Throughput	Gain
Soft SHA-256	295,860,775	2.959 s	0.95 MB/s	ref.
Hard SHA-256	38,376,545	0.384 s	7.29 MB/s	x7.7
+ DMA transfer	4,221,304	0.042 s	66.67 MB/s	x70

(a) SHA-256

IP	# Cycles	Boot time Overhead	Throughput	Gain
RSA-1024	92,867	0.001 s	N/A	N/A

(b) RSA-1024

Table 3.1: Performance overhead for the V6 VLX240T device

throughput with the flexible strategy using hardware RSA-1024 and SHA-256 with DMA transfers is 12.6 MB/s. That is faster than the classical Xilinx boot that does not use DMA transfers.

Considering only the integrity verification, the throughput is limited by the SHA-256 core and not the DMA data transfer. The hardware acceleration offers a high improvement (x70) and the total performance overhead is lower than 50 ms. These performances are really acceptable especially for a kernel boot context: that does not occur really often during application lifetime and it is not significant face to classical boot duration.

In this global loader architecture, the bottleneck remains the Flash to RAM copy due to the external Flash memory performances.

3.4.5.2 Area overhead

This secure solution without asymmetric cryptography and hardware acceleration can be implemented in software. Therefore, it is implemented without area overhead, except the code size overhead due to the software SHA-256 function.

The table 3.2 shows the area overhead with different strategies with regard to the base system. This four strategies match with the four strategies of the tables 3.1(a) and 3.1(b). For each strategy, the table 3.2 presents two results:

- In "Details" column, the area overhead details of the components added to the previous strategy.
- In "Total" column, the total area of the strategy.

To summarize, each performance or flexibility improvement occupies 1% more of the Virtex 6 VLX240T.

	Details				Total			
Strategies	**Components**	**Slice FF**	**Slice LUT**	**BRAM**	**Slice FF**	**Slice LUT**	**BRAM**	***Fraction of V6 VLX240T***
Base system (or with soft SHA-256)	MicroBlaze	3,196	3,874	19				
	Cache	6	14	16				
	DDR3	5,091	4,245	11				
	Flash	479	389					
	PLB	178	657		***8,950***	***9,179***	***46***	***6% + 11% de BRAM***
+ Hard SHA-256	SHA (+wrapper)	1,509	1,897	1				
	Interrupt ctrl.	190	180		***10,649***	***11,256***		***7% + 11% de BRAM***
+ DMA	Central DMA	561	799		***11,210***	***12,055***	***47***	***8% + 11% de BRAM***
+ RSA-1024	RSA (+wrapper)	684	989	4	***11,894***	***13,044***	***51***	***9% + 12% de BRAM***

Table 3.2: Area overhead for the V6 VLX240T device

However, all the components added in order to secure the boot can be reused for a targeted application. Moreover, the DMA controller is generally already present in most solutions.

Tables 3.1(a), 3.1(b) and 3.2 help to choose the adapted trade-off between timing and area overhead.

In a nutshell, the area overhead is negligible considering actual system or reusability.

3.4.6 Discussion

In this section, we proposed to secure the boot of a Linux embedded on a FPGA. The work presented here precludes malicious kernel modifications, prevents against replay attacks and supports updates. It highlights also the performance improvement of this security mechanism thanks to an FPGA capability: the hardware acceleration. ToIt also describes the implementation and evaluates the solution performance and area overhead.

This security enhancement will become increasingly useful with the FPGA evolution. For instance when FPGA vendor will provide hardly implemented mechanisms ensuring both confidentiality and integrity and preventing replay attacks.

In conclusion, this work shows that it is possible to secure efficiently the boot and that the hardware acceleration capabilities of FPGAs can be exploited to match the needs of embedded systems.

3.5 Conclusion

Securing embedded systems is a challenge requiring to investigate a wide range of domains. The mobility feature exposes the system with a larger attack surface in a further hostile environment. Moreover, it is accentuated with FPGAs due to the reconfiguration capabilities. That is why securing should impact every level from hardware to final application execution. Indeed, it is useless to secure an application if it runs on an untrusted hardware. Only one security flaw situated at any level, can turn the system hacked. In a global solution each securing step allows trusting a piece of architecture enable to secure the next step, establishing a end-to-end chain of trust.

As the previous chapter, this one also highlights the need to embedded flash memory inside the FPGA chip for security solutions in order to prevent software replay attacks, in addition to bitstream replay.

With Virtex 7, Xilinx proposes a multi-die technology in [Dorsey 2010]. The fact to embed different dies in a unique chip will be a good way to secure FPGA with

low over-costs. More generally, the democratization of SiPs (Systems in Package) will facilitate the production of FPGA with embedding flash memory.

Chapter 4

SecReSoC Platform

An idea is like a virus. Resilient. Highly contagious.
And even the smallest seed of an idea can grow.

Coob - Inception

Contents

This chapter presents the SecReSoC project supported and granted by the ANR -French National Research Agency- ARPEGE 2009 program (ANR-09-SEGI-013). It described the works that contributed to the achievement of this ambitious project. It replaces also the work presented in this dissertation into a shared project that aims to offer a complete protection for multiprocessor architectures based on FPGAs.

4.1 Project description

SecReSoC is an industrial research project that brings together the expertise of four academic laboratories with complementary skills: the Lab-STICC (Lorient), the LAHC (Saint Etienne), the LIRMM (Montpellier) and Télécom ParisTech (Paris) and a Company recognized in the cryptography domain: Netheos (Montpellier).

The objective of this project is to develop a generic multiprocessor architecture that allows the integration, into an FPGA platform, of an application requiring different levels of data and treatment securing (logical, architectural and system level). Indeed, FPGAs are essential to prototype cryptographic architectures and their reconfiguration capability allows them to evolve in order to prevent vulnerabilities and counter new attacks. Moreover, the complexity of current applications justifies multiprocessor solutions.

This architecture will include an optimized cryptoprocessor, some standard processors associated with a multitasking OS environment, memory resources, input/output units and an internal architecture of communication. Secure configuration tools and different types of physical attacks are also studied.

All confidential treatments and handling of critical plaintext data are performed within the security perimeter, which is located inside the reconfigurable circuit. This is the main constraint of the project which will result in the development of a demonstrator. The SecReSoC project meets the growing needs in integration and securing of applications requiring autonomous devices capable of operating in hostile environments.

The SecReSoC project provides a set of software and hardware blocks (Firewalls, HCrypt, DPA counter-measures, secure reconfiguration, monitoring, cryptocores and key management) designed to match the interests of the industrial world. Demonstrators use these blocks to address different threat models.

The goal is to provide a platform proving the concepts involved to ensure:

- Security: by architecture design, the partitioning precludes leakages and the hardware counter-measures.
- Performances: by implementing hardware acceleration thanks to FPGAs abilities.
- Ease-of-use: by offering the possibility to use un standard OS (Linux) that proposes a complete set of features for non-critical issues and by proposing full-compliant HDL modules (interfaced with the AXI standard bus).

The threat model considers that:

- The attacker has read and write access to board memories.

- The attacker may attempt to copy or counterfeit an equipment.
- All interfaces (eg: network) excepting interfaces considered sure (eg: Safepad when user is present) may be hacked.
- On-line takeover of the GPP (General Purpose Processor).
- Takeover of the bus system through the GPP.
- Side-channel attacks (SEMA, DEMA, DPA and DFA (Differential Fault Analysis)) are feasible.

However, invasive attacks are not considered.
Figure 4.1 is an overview of the SecReSoC module composed of:

- A trusted MicroBlaze responsible for hardware and software updates, board monitoring, trusted mode execution.
- Firewalls to ensure partitioning
- A safepad with a smart card in charge of keys initialization.
- Cryptocores to perform hardware acceleration.

Sensible data are protected thanks different keys listed in Table 4.1:

Keys	ID	Type	Owner	Handled by the GPP	Handled by the SecReSoC module	Storage
Bitstream key	K_{B}	Symmetric	Vendor	No	No	FPGA NVM
Vendor's key	Ks_{Vendor}	Symmetric	Vendor	No	Yes	Bitstream
Vendor's public key	Kp_{Vendor}	Asymmetric	Vendor	Yes	Yes	Bitstream
Identification key	K_{ID}	Asymmetric	User	No	Yes	External NVM (ciphered by Ks_{Vendor})
Personalization key	K_{Perso}	Symmetric	User	No	Yes	External NVM (ciphered by Ks_{Vendor})
Operating keys		Symmetric or Asymmetric	User	Yes, by reference	Yes	External NVM (ciphered by Ks_{Perso})

Table 4.1: Details of the involved keys

4.2 Details of the SecReSoC couter-measures

The SecReSoC project is the gathering of several works detailed in this section. In addition to the work carried out during this thesis, others works have contributed to the project.

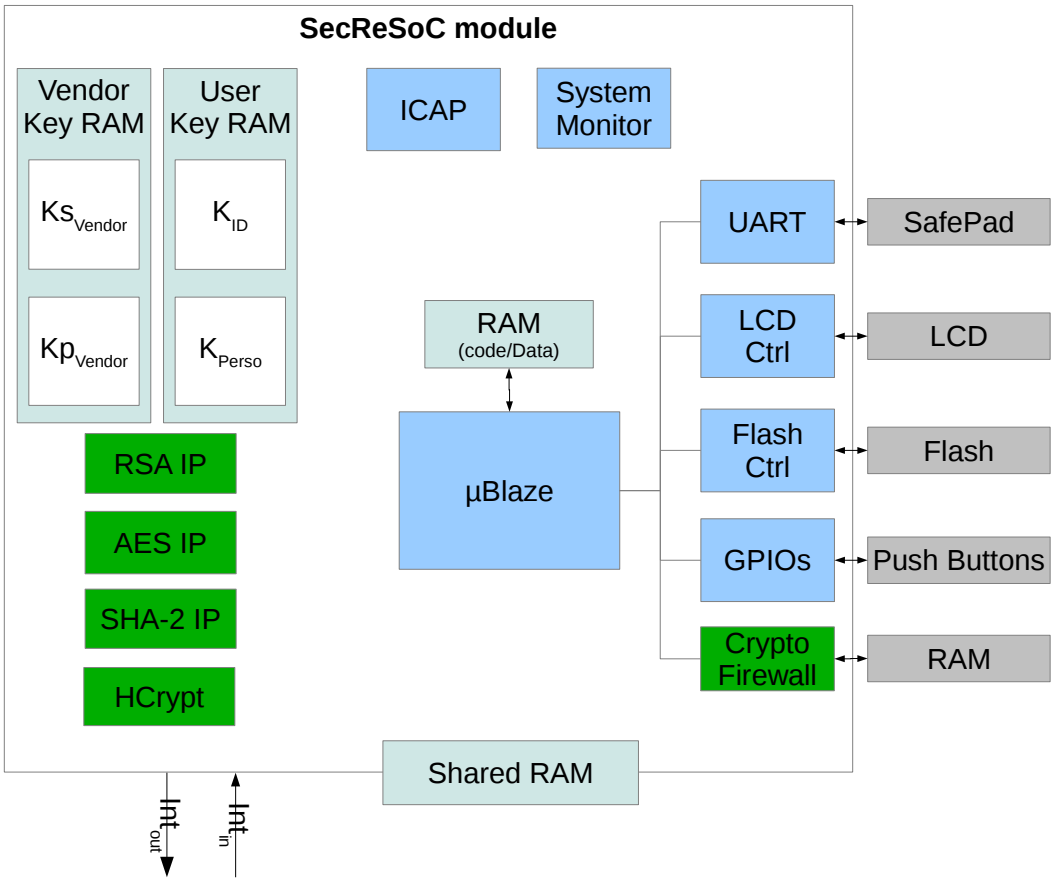


Figure 4.1: SecReSoC module overview

4.2.1 Hardware Firewalls

Hardware firewalls [Cotret 2011] principle is based on isolation. Figure 4.2 shows that firewalls are placed in cut-through configuration at each bus/IPs and bus/processors connection. Firewalls aim to monitor communications before they reach the bus and propagate within the system.

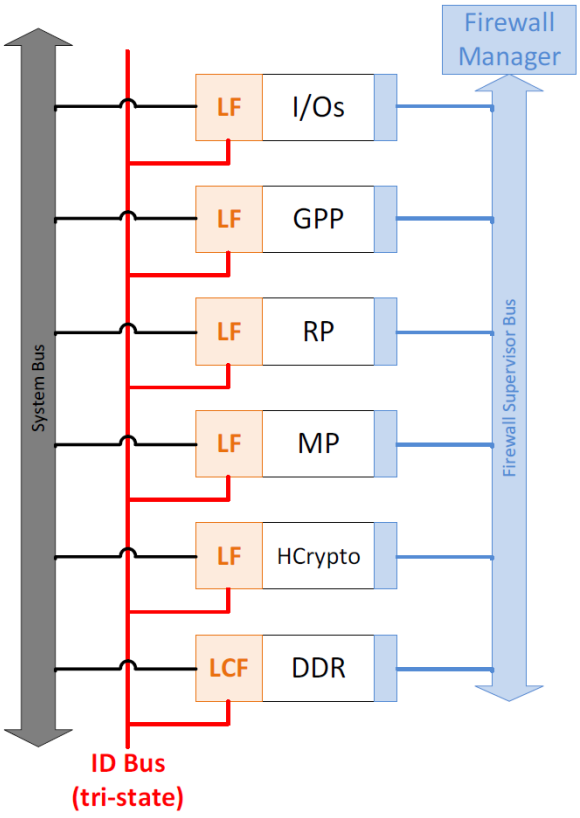


Figure 4.2: Firewalls outline in a complete architecture

As shown in Figure 4.3, there are two categories of firewalls: Local Firewalls and a Cryptographic Firewall.

Local Firewalls restrict the access rights block by block and allow two configurations: normal and restricted. Interruptions are raised in case of attempt to prohibited access. All these attempts are logged in a file.

The Cryptographic Firewall is a Local Firewall enhanced with a layer of cryptographic services like memory encryption with AES-GCM. It is located towards the external DDR memory in order to protect it. Two zones are defined with different access rights and cryptographic counter-measures (encryption and integrity verification or integrity verification only). The goal is to have several options in order to get the better security/performances trade-off. A new symmetric key is generated

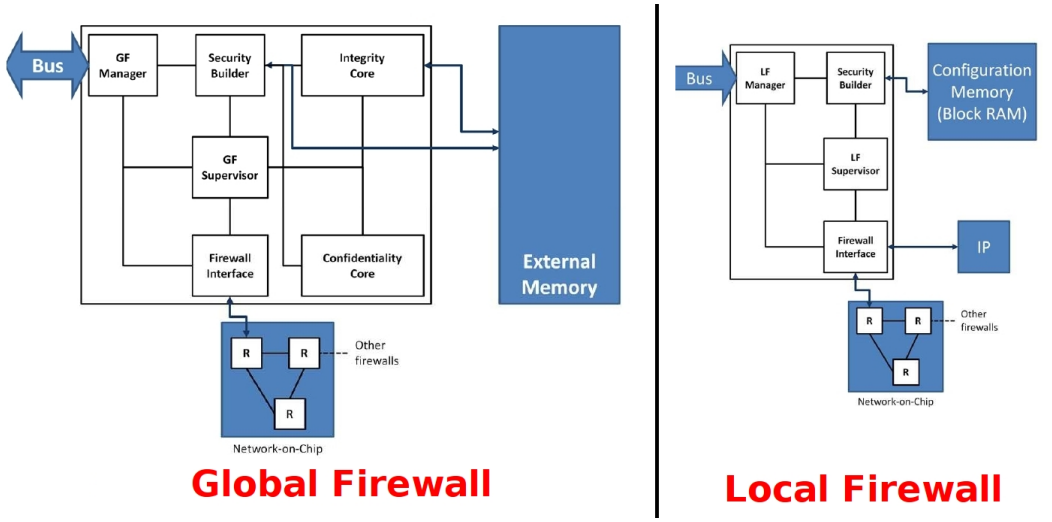


Figure 4.3: Local and Cryptographic Firewalls

on every boot to preclude side-channel attacks.

4.2.2 HCrypt: the cryptoprocessor

HCrypt [Gaspar 2010] is developed to achieve physical isolation of data, secret key management and buses techniques preventing leaking information. Its architecture is optimized for implementation of common cryptography tasks.

Figure 4.4 is an overview of the HCrypt. This cryptoprocessor has 128-bit separated data and key registers, dedicated instruction set optimized for key generation and management, embedded cipher, and embedded random number generator. From an architectural point of view, the most important characteristic of the proposed cryptoprocessor is the physical separation of data and key registers and buses, insuring that confidential keys will never leave the system in clear.

Figures 4.5(a) and 4.5(b) represent the different frame types that are involved in the HCrypt protocol.

4.2.3 Side-channel counter-measures

This work proposes a new concept [Maghrebi 2011] to hinder attacks of all order: instead of injecting more entropy, it makes the most of a single-mask entropy. With specially crafted bijections instantiated on the mask path, it manages to reduce the inter-class variance (a method called *leakage squeezing*) so that the leakage distributions become almost independent from the processed data. This contribution presents two options for this counter-measure. The first one is based on a recoded memory with a size squared with regard to the unprotected requirement, whilst the second one is an enhancement alleviating the requirement for a large memory.

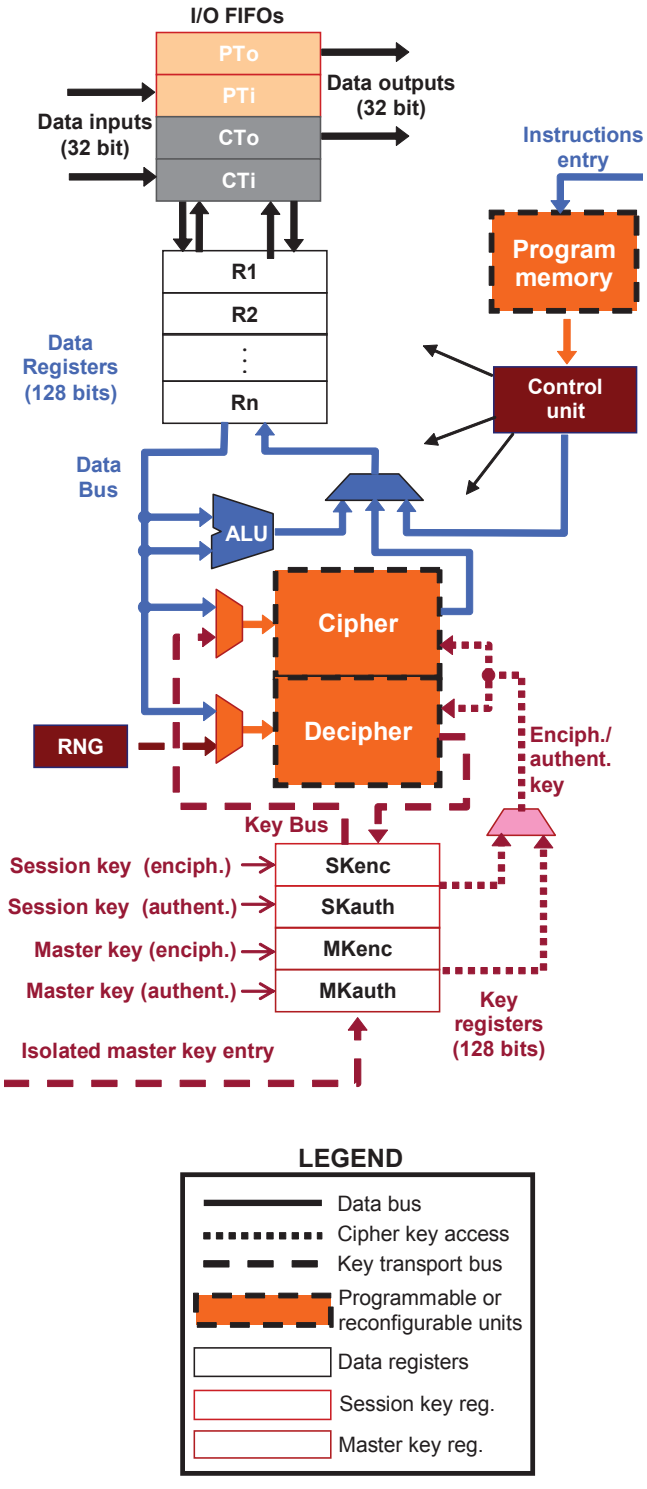


Figure 4.4: The HCrypt architecture

Decryption request	Header	Command	Size	Session key	Signature of the session key	Packet (ciphered)	End of frame
Encryption request	Header	Command	Size	N/A	N/A	Packet (plaintext)	End of frame

(a) HCrypt request frames (from GPP to HCrypt)

Decryption reply	Header	Status	Size	N/A	N/A	Packet (plaintext)	End of frame
Encryption reply	Header	Status	Size	Session key	Signature of the session key	Packet (ciphered)	End of frame

(b) HCrypt reply frames (from HCrypt to GPP)

Figure 4.5: Frame types involved in the HCrypt protocol

4.3 Demonstrators

The solutions presented in this dissertation and in the previous section are used to produce several demonstrators. Each of them implements several counter-measures in order to prove their feasibility and efficiency. Those demonstrators are based on the Xilinx ML605 development board embedding a Virtex-6 XC6VLX240T shown in Figure 4.6.

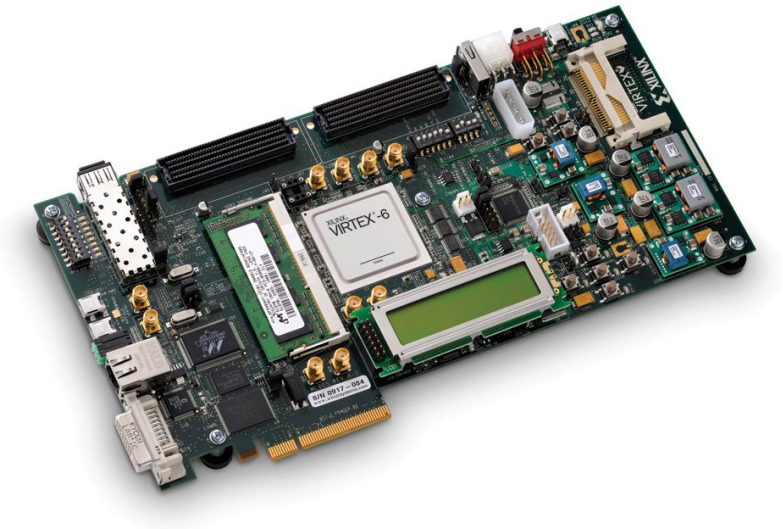


Figure 4.6: The Xilinx ML605 development board

All the versions and applications of the SecReSoC demonstrator are protected thanks to the work presented in this thesis. Moreover, the design of these demonstrators is an important part of the work accomplished during my thesis. Indeed, the achievement of the demonstrators was assumed by my thesis work.

4.3.1 Version 1: Embedded Crypto-Signer

This first version of the demonstrator is a Linux server able to encrypt and sign a file or decrypt and verify a file signature.

It operates as follows:

- Encryption / signature: The encryption operation is done simply by depositing a file in a directory of the FTP (File Transfer Protocol) server's device (eg: ftp://ECS/toCrypt). This operation does not require special authorization (no authentication or PIN (Personal Identification Number) code). The ECS (Embedded Crypto-Signer) creates the encrypted file with its signature in a particular FTP directory (eg: ftp://ECS/encrypted). To select a recipient's public key in the certificate store of the ECS, files to be encrypted contain a header describing the recipient.

- Encryption / signature verification: The decryption operation is similar. The encrypted file is placed in the good folder (eg: ftp://ECS/toDecrypt). The ECS generates the decrypted file in the appropriate FTP folder according to the result of the signature verification (eg: ftp://ECS/decrypted_signOK or ftp://ECS/decrypted_signFail).

Figure 4.7 shows an overview of the SecResoC demonstrator version 1, whilst Figure 4.8 shows a screenshot of the application being run.

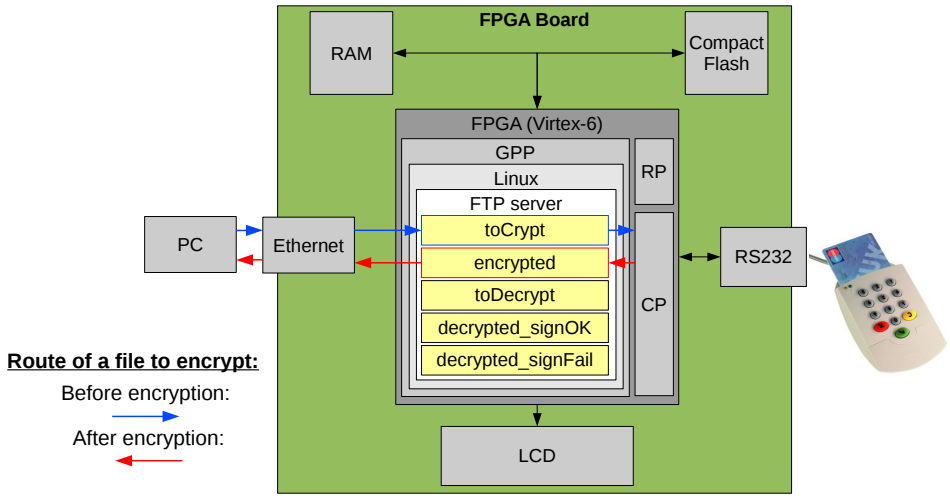


Figure 4.7: Overview of the SecResoC demonstrator

This version implements the HCrypt version 1.27 and uses a Xilinx architecture with a PLB bus. The FTP server is a vsFTPd (Very Secure FTP Daemon) [vsf] running on Xilinx embedded Linux [Xilinx]. Figure 4.9 is an overview of the internal architecture of the SecResoC demonstrator version 1.

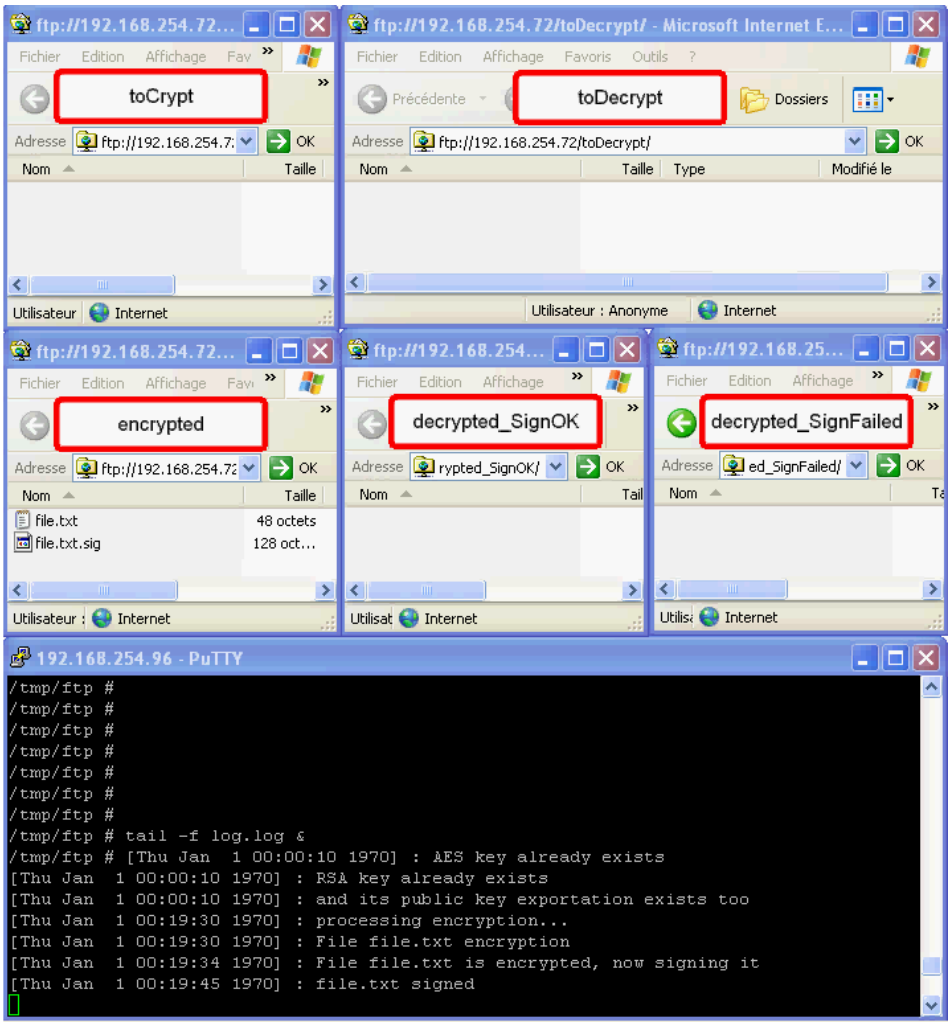


Figure 4.8: Screenshot of the ECS application

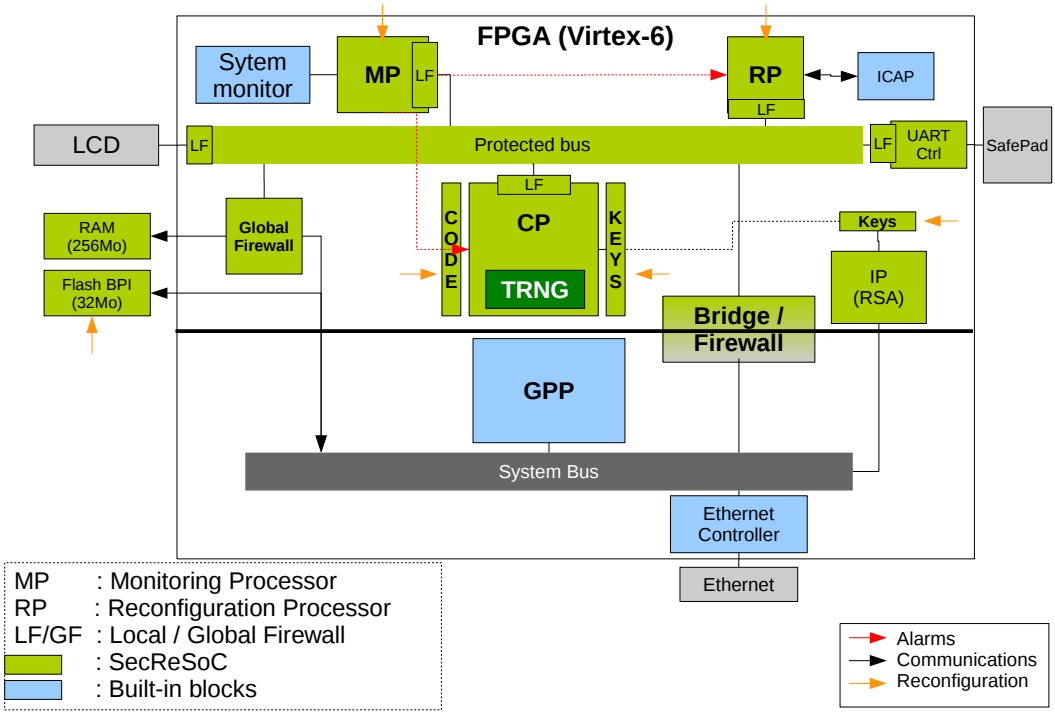


Figure 4.9: Overview of the SecResoC demonstrator version 1 architecture

This demonstrator is a progress step that proves it is possible to have such architecture before put together the different works. This prototype achievement is an integral part of this thesis. For instance, my work was to protect the Linux boot and to establish of the FTP server as well as all the scripts able to provide the desired functionalities.

4.3.2 Version 2: Several applications

The second version of the SecReSoC demonstrator differs from the first by its bus. The change on Xilinx bus standard, have an impact on our demonstrator. Indeed, the PLB bus is replaced by an AXI one. Work progresses made possible integrate others counter-measures like illustrated in Figure 4.10. The work achieved in the frame of this thesis was the integration of IPs and counter-measures for PLB bus, then for AXI bus. My work consisted in porting the Linux on this new AXI bus architecture.

4.3.2.1 Application 1: Low-cost Embedded Crypto-Signer

This application is the extension of the version 1. A new version of the HCrypt (version 2.00) embedding side-channel counter-measures explained in section 4.2.3. In addition the PRNG (Pseudo-Random Number Generator) is replaced by a TRNG

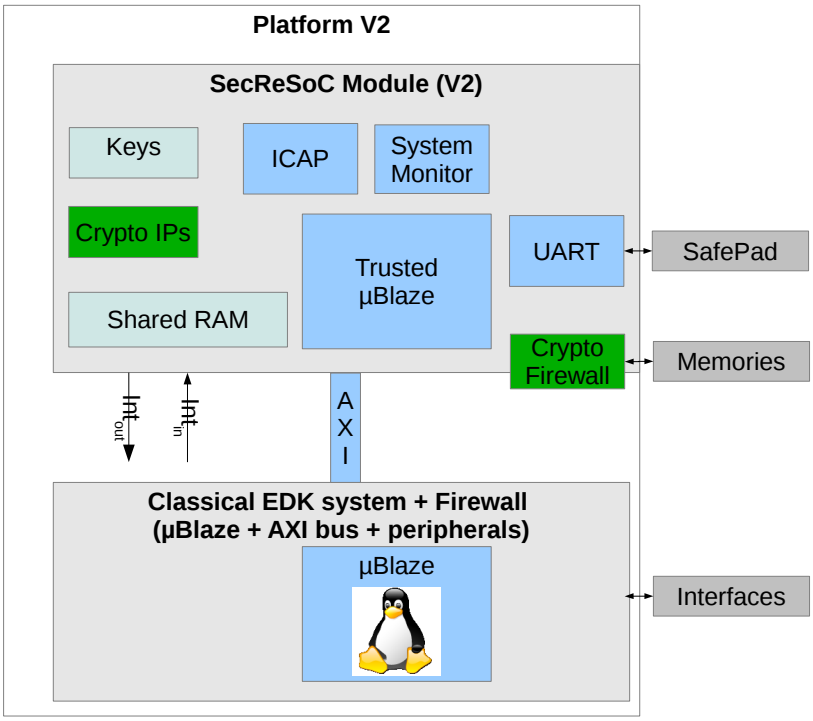


Figure 4.10: Overview of the SecResoC demonstrator version 2

(True Random Number Generator) based on PLLs (Phase-Locked Loops). This new version of the HCrypt is integrated and interfaced with the new AXI environment (see Figure 4.11).

The usage is the same as in the previous version.

4.3.2.2 Application 2: Advanced Crypto-Signer

Figure 4.12 shows that the main concept of the ACS (Advanced Crypto-Signer) is a separation in two distinct worlds:

- A generalist one (GPP), highly exposed (eg: network), running a common OS with embedding many features.
- And a secure one (TrustedBlaze), isolated, running a trusted OS or a program, in charge of critical treatments.

Figure 4.13 is an example of use case. The user browses the web until he wants to make a critical operation in terms of security (eg: a purchase, a bank transfer, etc...). At that time, the TrustedBlaze takes control of the platform to perform this critical operation.

Here, the use of HCrypt is not required since the isolation is ensured by the hardware firewalls represented by the switch in Figure 4.13. This switch connects

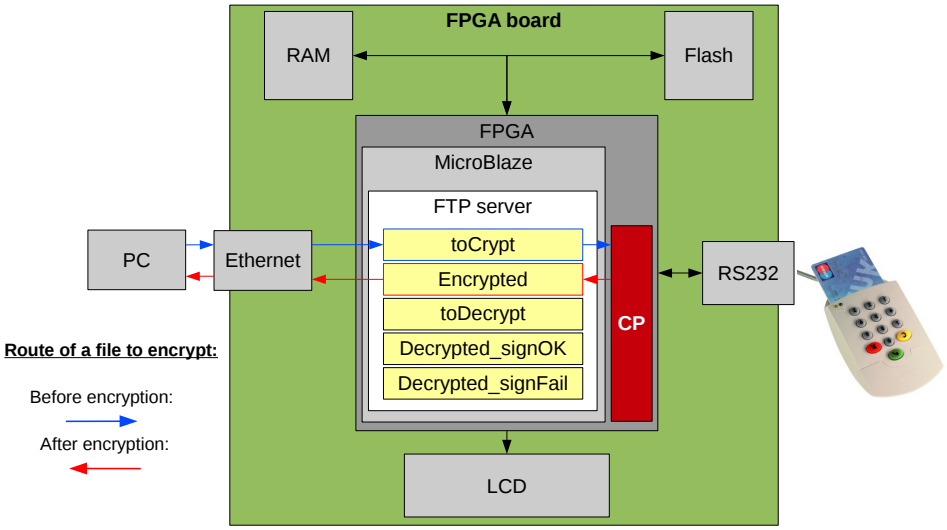


Figure 4.11: Low-cost ECS overview

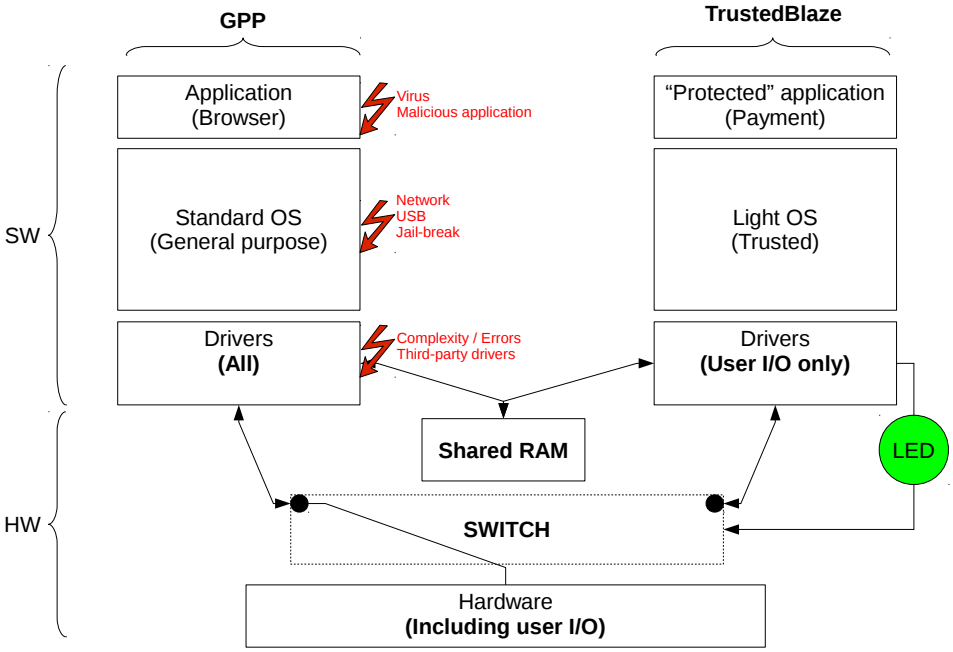


Figure 4.12: ACS overview

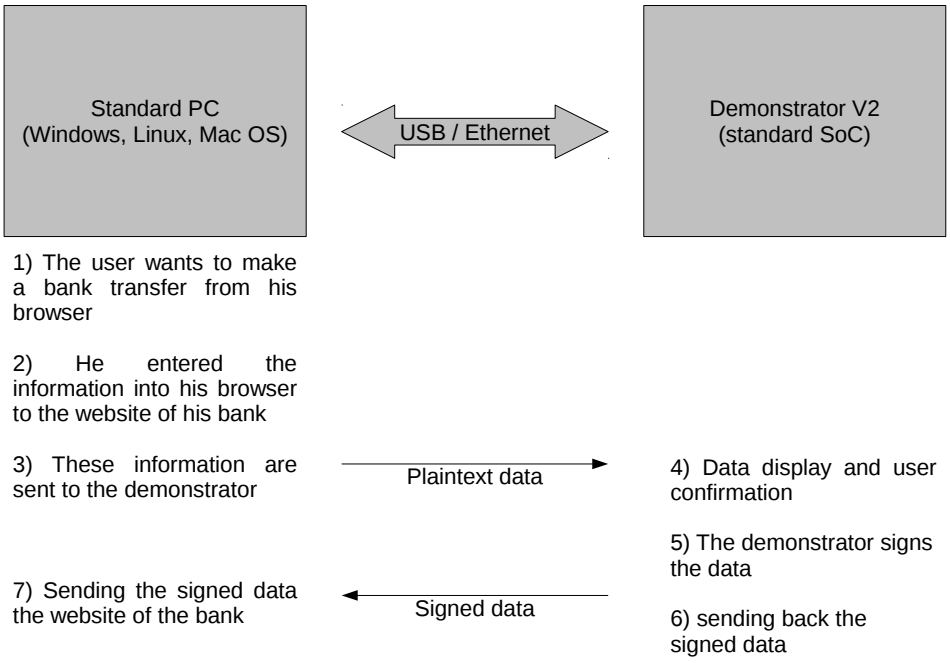


Figure 4.13: ACS use case

the user I/O to the GPP or to the TrustedBlaze (the latter controls the switch). The LED driven by the TrustedBlaze certifies that the demonstrator is running in restricted mode.

4.3.2.3 Application 3: Reconfigurable Crypto-Signer

The RCS (Reconfigurable Crypto-Signer) is based on the ECS (version 2, application 1). The difference lies in the addition of the DPR support and protection. The DPR aims to reduce the power consumption and the area of the design. For instance, it is possible to reconfigure the cryptocores (eg: switch between SHA256 and SHA512). The contribution to this application corresponds to the Section 2.3.

4.4 Conclusion

Table 4.2 highlights the solutions implemented and the threats addressed by each demonstrator.

The SecReSoC project allows sharing the work of several laboratories in order to have a common platform that offers effective solutions to a complete threats model. It also helps to implement the work carried out in this thesis a platform to gain greater visibility. The design of these demonstrators brought me a lot in terms of knowledge and skills about embedded systems and OSs based on FPGAs.

Demonstrator	Threats							
	Boot attacks	Non Volatile Memory access	Copy / Counterfeit	Taking control of the GPP	Man-in-the-system attacks	Side-channel attacks	Partial bitstreams attacks
ECS (application 1)	X	X	X	X	X		
ACS (application 2)	X	X	X	X	X	X	
RCS (application 3)	X	X	X	X	X	X	X

Corresponding solutions					
System integrity Restricted mode	Key management	Identification key	Partitioning (firewalls, keys)	I / O isolation Restricted mode	HCrypt Counter-measures	Partial bitstreams protection

Table 4.2: Solutions to the threats by demonstrator

Conclusion

This dissertation presents a research work that has been conducted in order to increase the security level of embedded systems based on FPGA technologies. The objective was to protect the embedded system itself and its sensitive data by preventing an attacker that would spy data transfers in order to retrieve information or insert malicious code. This work addresses more particularly replay attacks that are not currently considered by FPGA vendors. This lack exposes embedded systems to critical vulnerabilities.

Contributions

This thesis provides a large state of the art of attacks applicable to FPGA devices.

This dissertation addresses the issues presented in the introduction: it is to maintain confidentiality, integrity and authenticity while providing up-to-dateness. This thesis provides dedicated and suitable solutions to these problems by considering the entire lifecycle of the embedded system (boot, updates and code or OS execution) and all the data (FPGA bitstream, operating system kernel, critical data and code).

This work addresses all the current FPGA technologies (Flash and SRAM) taking into account their strengths and weaknesses. It covers a very large domain application by considering advanced FPGA capabilities like DPR. Most of the solutions presented in this document have been implemented to prove the feasibility as well as to evaluate area and performance overheads. They are very efficient by taking advantage of hardware acceleration offered by FPGA capabilities in order to match the expectations of embedded systems.

Analyses

This work is conducted step by step, each securing step allowing to trust a piece of architecture enable to secure the next step, establishing a end-to-end chain of trust. It highlights that it is important to consider the security in its entirety. That is why this dissertation attempts to address every level from hardware to final application execution. Indeed, it is important to analyze the threat model and not to forget the slightest security flaw situated at any level from hardware to software that can turn the system hacked.

Side channel attacks are deliberately not addressed in this work but should be considered. In fact, embedded systems are particularly exposed to such attacks.

Side channel counter-measures are very advanced, and rely on complex mechanisms, but it is important to preclude at least timing attack and to periodically change encryption keys. This aims to increase the security level even if all attacks are not counteracted.

This work show also that the security mechanisms provided by FPGA vendors are effective but insufficient. Nowadays academic literature offers solutions ready to be implemented but still not yet deployed by FPGA manufacturers. This can be explained by the fact that FPGA vendors must find a good trade-off between functionalities and price of their chips. In addition, it is difficult to offer many specific devices corresponding to several uses. Indeed, the concept brought by FPGA is to have a limited number of different chips in order reduce prices thanks to economies of scale.

Perspectives

The short-term work will be to finalize the sharing of the four thesis works conducted in the frame of the SecReSoC project. Indeed, the several demonstrators will be completed in order to highlight the efficiency of the contributions provided by this joint work of several laboratories and a company.

As presented in Chapter 4, the demonstrators are based on industrially-driven applications. These applications allow demonstrating the maturity of the project and the efficiency of counter-measures introduced by the different works.

The work based on the security of embedded systems based on FPGA devices is not over. It has contributed to the creation of a company: SECLAB FR.

The best fulfillment for this work would be to promote FPGA vendors to introduce the work carried out here in their products.

Publications Relative to the Study

Book Chapter

- Benoît Badrignans, Jean-Luc Danger, Viktor Fischer, Guy Gogniat and Lionel Torres. ***Security Trends for FPGAS: From Secured to Secure Reconfigurable Systems***, Chapter 6, available at: http://www.springer.com/engineering/circuits+%26+systems/book/978-94-007-1337-6

International conference papers

- Florian Devic, Lionel Torres and Benoît Badrignans ***Secure protocol implementation for remote bitstream update preventing replay attacks on FPGA*** In FPL 2010: 20th IEEE International Conference on Field Programmable Logic and Applications, pages 179-182, Milano, Italia, September 2010. available at: http://dx.doi.org/10.1109/FPL.2010.44

- Florian Devic, Lionel Torres and Benoît Badrignans ***Securing Boot of an Embedded Linux on FPGA*** In IPDPS/RAW 2011: 25th IEEE International Symposium on Parallel & Distributed Processing / 18th Reconfigurable Architectures International Workshop, Anchorage, Alaska, USA, May 2011. available at: http://dx.doi.org/10.1109/IPDPS.2011.141

National conference papers

- Florian Devic, Lionel Torres and Benoît Badrignans ***Secure protocol implementation for remote bitstream update preventing replay attacks on FPGA*** In GDR SoC-SiP 2010: Groupement De Recherche System on Chip System in Package, Paris, France, June 2010.

- Florian Devic, Lionel Torres and Benoît Badrignans ***Remote bitstream update protocol preventing replay attacks: A practical implementation*** In Cryptarchi 2010: 8th Cryptographic Architectures Embedded in Reconfigurable Devices International Workshop, Gif-sur-Yvette, France, June 2010.

- Florian Devic, Lionel Torres and Benoît Badrignans ***Embedded OS boot protection for FPGA platforms*** In GDR SoC-SiP 2011: Groupement De Recherche System on Chip System in Package, Lyon, France, June 2011.

- Florian Devic, Lionel Torres and Benoît Badrignans ***Embedded OS for FPGA platform: a Hardware-to-Software Security Overview*** In Cryptarchi 2011 : 9th Cryptographic Architectures Embedded in Reconfigurable Devices International Workshop, pages 189-195, Bochum, Germany, June 2011.

Bibliography

[Abraham 1991] D. G. Abraham, G. M. Dolan, G. P. Double and J. V. Stevens. *Transaction security system*. IBM Syst. J., vol. 30, pages 206–229, March 1991, available at: http://dx.doi.org/10.1147/sj.302.0206. (Cited on page 7.)

[Ahmed 2011] Syed Zahid Ahmed. *EFPGAs : Architectural Explorations, System Integration, & a Visionary Industrial Survey of Programmable Technologies*. PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, June 2011. (Cited on page 15.)

[Altera] Altera corporation. *Military Anti-Tampering Solution with High-Performance Stratix II and Stratix II GX FPGAs*, available at: http://www.altera.com/products/devices/stratix-fpgas/stratix-ii/stratix-ii/features/security/st2-security-anti-tamper.html. (Cited on page 29.)

[Altera 2009] Altera corporation. *Remote System Upgrade (ALTREMOTE_UPDATE): Megafunction User Guide*, April 2009. Version 2.4, available at: http://www.altera.com/literature/ug/ug_altremote.pdf. (Cited on page 31.)

[AMD 2005] AMD corporation. *AMD64 Virtualization Technology: Secure Virtual Machine Architecture Reference Manual*, December 2005. Publication number 33047, Revision 3.02, available at: http://www.0x04.net/doc/amd/33047.pdf. (Cited on page 60.)

[AMD 2009] AMD corporation. *AMD I/O Virtualization Technology (IOMMU) Specification*, February 2009. Revision 1.6, PID: 34434, available at: http://support.amd.com/us/Processor_TechDocs/34434-IOMMU-Rev_1.26_2-11-09.pdf. (Cited on page 60.)

[ARM 2009] ARM corporation. *Building a Secure System using TrustZone Technology*, April 2009. white paper, available at: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf. (Cited on pages 61 and 64.)

[Atmel 2006] Atmel corporation. *Safe and Secure Bootloader Implementation*, December 2006. literature no. 6282, available at: http://www.atmel.com/dyn/resources/prod_documents/doc6253.pdf. (Cited on pages 64 and 65.)

[Badrignans 2008] Benoît Badrignans, Reouven Elbaz and Lionel Torres. *Secure FPGA Configuration Architecture Preventing System Downgrade.* In FPL'08: IEEE International Conference on Field Programmable Logic and Applications, pages 317–322, Heidelberg, Germany, September 2008, available at: http://dx.doi.org/10.1109/FPL.2008.4629951. (Cited on pages 28, 38 and 44.)

[Badrignans 2009] Benoît Badrignans. *Using FPGAs for security-sensitive applications.* PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, December 2009. (Cited on pages 31 and 32.)

[Badrignans 2010] Benoît Badrignans, David Champagne, Reouven Elbaz, Catherine Gebotys and Lionel Torres. *SARFUM: Security Architecture for Remote FPGA Update and Monitoring.* ACM Trans. Reconfigurable Technol. Syst., vol. 3, pages 8:1–8:29, May 2010, available at: http://doi.acm.org/10.1145/1754386.1754389. (Cited on pages 63, 64 and 69.)

[Bläsing 2010] Thomas Bläsing, Aubrey-Derrick Schmidt, Leonid Batyuk, Seyit A. Camtepe and Sahin Albayrak. *An Android Application Sandbox System for Suspicious Software Detection.* In 5th International Conference on Malicious and Unwanted Software (MALWARE'2010), Nancy, France, 2010, available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5665792. (Cited on page 59.)

[Bossuet 2004] Lilian Bossuet, Guy Gogniat and Wayne Burleson. *Dynamically Configurable Security for SRAM FPGA Bitstreams.* volume 4, page 146, Santa Fe, New Mexico, April 2004, available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1303128. (Cited on page 43.)

[Braeken 2011] An Braeken, Jan Genoe, Serge Kubera, Nele Mentens, Abdellah Touhafi, Ingrid Verbauwhede, Yannick Verbelen, Jo Vliegenyz and Karel Wouters. *Secure Remote Reconfiguration of an FPGA-based Embedded System.* In ReCoSoC 2011 (Reconfigurable and Communication-centric Systems-on-Chip), 2011. (Cited on page 31.)

[CC 2009] *Common Criteria for Information Technology Security Evaluation*, july 2009. version 3.1 revision 3, available at: http://www.ssi.gouv.fr/fr/certification-qualification/cc/les-criteres-et-methodologies-d-evaluation.html. (Cited on page 9.)

[Chawla 2009] Kirti Chawla and Taniya Siddiqua. *Mutation Resistant Runtime Code using Kernel Attestation*, 2009. OSSEC project, available at: http://www.cs.virginia.edu/~kc5dm/projects/ossec.pdf. (Cited on page 31.)

[Cotret 2011] Pascal Cotret, Jeremie Crenne, Guy Gogniat, Jean-Philippe Diguet, Lubos Gaspar and Guillaume Duc. *Distributed Security for Communications and Memories in a Multiprocessor Architecture*. In 18th Reconfigurable Architectures Workshop (RAW/IPDPS), Anchorage, Alaska, USA, May 2011, available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6008915. (Cited on pages 58 and 79.)

[Crenne 2011] Jérémie Crenne, Romain Vaslin, Guy Gogniat, Jean-philippe Diguet, Russell Tessier and Deepak Unnikrishnan. *Configurable Memory Security In Embedded Systems*, 2011, available at: http://www.ecs.umass.edu/ece/tessier/crenne-tecs.pdf. (Cited on page 57.)

[de Cesare 2009] Joshua de Cesare, October 2009. US 2009/0257595, Apple corporation, available at: http://www.freepatentsonline.com/20090257595.pdf. (Cited on page 64.)

[Devic 2010] Florian Devic, Benoît Badrignans and Lionel Torres. *Secure Protocol Implementation for Remote Bitstream Update Preventing Replay Attacks on FPGA*. In FPL'10: IEEE International Conference on Field Programmable Logic and Applications, pages 179–182, Milano, Italia, September 2010, available at: http://dx.doi.org/10.1109/FPL.2010.44. (Cited on pages 44, 63 and 64.)

[Discretix] Discretix corporation. *Discretix Secure Boot (DxSB)*, available at: http://www.discretix.com/secureboot/index.html. (Cited on page 64.)

[Dorsey 2010] Patrick Dorsey. *Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency*, 2010. Xilinx corporation, available at: http://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf. (Cited on page 73.)

[Drimer 2009] Saar Drimer and Markus Kuhn. *A Protocol for Secure Remote Updates of FPGA Configurations*. In Jürgen Becker, Roger Woods, Peter Athanas and Fearghal Morgan, editeurs, Reconfigurable Computing: Architectures, Tools and Applications, volume 5453 of *Lecture Notes in Computer Science*, pages 50–61. Springer Berlin / Heidelberg, 2009, available at: http://dx.doi.org/10.1007/978-3-642-00641-8_8. (Cited on pages 31, 36 and 44.)

[Eisenbarth 2007] Thomas Eisenbarth, Tim Güneysu, Christof Paar, Ahmad-Reza Sadeghi, Dries Schellekens and Marko Wolf. *Reconfigurable trusted computing in hardware*. In Proceedings of the 2007 ACM workshop on Scalable trusted

computing, STC '07, pages 15–20, New York, NY, USA, 2007. ACM, available at: http://doi.acm.org/10.1145/1314354.1314360. (Cited on page 58.)

[Elbaz 2006] Reouven Elbaz, Lionel Torres, Gilles Sassatelli, Pierre Guillemin, Michel Bardouillet and Albert Martinez. *A parallelized way to provide data encryption and integrity checking on a processor-memory bus.* In Proceedings of the 43rd annual Design Automation Conference, DAC '06, pages 506–509, New York, NY, USA, 2006. ACM, available at: http://doi.acm.org/10.1145/1146909.1147042. (Cited on page 56.)

[Eldredge 1996] James G. Eldredge and Brad L. Hutchings. *Run-Time Reconfiguration: A method for enhancing the functional density of SRAM-based FPGAs.* The Journal of VLSI Signal Processing, vol. 12, pages 67–86, 1996. 10.1007/BF00936947, available at: http://dx.doi.org/10.1007/BF00936947. (Cited on page 41.)

[FIPS 2001] Donald L. Evans, Phillip J. Bond and Arden L. Bement. *Security Requirements For Cryptographic Modules*, may 2001. Information Technology Laboratory National Institute of Standards and Technology, Gaithersburg, MD 20899-8900, (Supercedes FIPS PUB 140-1, 1994 January 11), available at: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf. (Cited on page 11.)

[Gaspar 2010] Lubos Gaspar, Viktor Fischer, Florent Bernard, Lilian Bossuet and Pascal Cotret. *HCrypt: A Novel Concept of Crypto-processor with Secured Key Management.* In Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs, RECONFIG '10, pages 280–285, Washington, DC, USA, 2010. IEEE Computer Society, available at: http://dx.doi.org/10.1109/ReConFig.2010.38. (Cited on page 80.)

[Halderman 2009] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum and Edward W. Felten. *Lest we remember: cold-boot attacks on encryption keys.* Commun. ACM, vol. 52, pages 91–98, May 2009, available at: http://doi.acm.org/10.1145/1506409.1506429. (Cited on page 23.)

[Höflich 1994] Wolfgang Höflich. *Using the XC4000 Readback Capability*, 1994. Applications Notes, XAPP 015.000, Xilinx corporation, available at: http://www.xilinx.com/support/documentation/application_notes/xapp015.pdf. (Cited on page 29.)

[Hill 2010] Brian Hill. *Embedded Platform Software and Hardware In-the-Field Upgrade Using Linux.* Xilinx corporation, May 2010. Application

Note: xapp1146, available at: www.xilinx.com/support/documentation/application_notes/xapp1146.pdf. (Cited on page 65.)

[Hori 2008] Yohei Hori, Akashi Satoh, Hirofumi Sakane and Kenji Toda. *Bitstream Encryption and Authentication Using AES-GCM in Dynamically Reconfigurable Systems*. In FPL'08: IEEE International Conference on Field Programmable Logic and Applications, pages 23–28, Heidelberg, Germany, September 2008, available at: http://dx.doi.org/10.1109/FPL.2008.4629902. (Cited on page 44.)

[Huang 2003] Andrew Huang. *Keeping Secrets in Hardware: The Microsoft Xbox ™ Case Study*. In Burton Kaliski, çetin Koç and Christof Paar, editeurs, Cryptographic Hardware and Embedded Systems - CHES 2002, volume 2523 of *Lecture Notes in Computer Science*, pages 355–430. Springer Berlin / Heidelberg, 2003, available at: http://dx.doi.org/10.1007/3-540-36400-5_17. (Cited on pages 25, 56 and 64.)

[IBM 2010] IBM corporation. *IBM X-Force®2010: Mid-Year Trend and Risk Report*, August 2010, available at: ftp://public.dhe.ibm.com/common/ssi/ecm/en/wgl03003usen/WGL03003USEN.PDF. (Cited on page 61.)

[Intel 2011] Intel corporation. *Intel® Virtualization Technology for Directed I/O*, February 2011. Revision 1.3, Order Number: D51397-005, available at: http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_IO.pdf. (Cited on page 60.)

[Kepa 2010] Krzysztof Kepa, Fearghal Morgan, Krzysztof Kosciuszkiewicz and Tomasz Surmacz. *SeReCon: a secure reconfiguration controller for self-reconfigurable systems*. Int. J. Crit. Comput.-Based Syst., vol. 1, pages 86–103, February 2010, available at: http://dx.doi.org/10.1504/IJCCBS.2010.031707. (Cited on page 44.)

[Keurentjes 2007] Niels Keurentjes. *Xbox 360 ran homebrew, leak patched*, February 2007, available at: http://www.xboxic.com/news/2485. (Cited on page 61.)

[Kocher 1996] Paul C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*. In Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO' 96, pages 104–113, London, UK, UK, 1996. Springer-Verlag, available at: http://portal.acm.org/citation.cfm?id=646761.706156. (Cited on pages 20 and 42.)

[Kocher 1998] Paul Kocher, Joshua Jaffe and Benjamin Jun. *Introduction to differential power analysis and related attacks*, 1998, available at:

http://www.cryptography.com/public/pdf/DPATechInfo.pdf. (Cited on page 20.)

[Kocher 1999] Paul Kocher, Joshua Jaffe and Benjamin Jun. *Differential Power Analysis*. In Michael Wiener, editeur, Advances in Cryptology - CRYPTO' 99, volume 1666 of *Lecture Notes in Computer Science*, pages 789–789. Springer Berlin / Heidelberg, 1999, available at: http://dx.doi.org/10.1007/3-540-48405-1_25. (Cited on page 21.)

[Kocher 2004] Paul Kocher, Ruby Lee, Gary McGraw and Anand Raghunathan. *Security as a new dimension in embedded system design*. In Proceedings of the 41st annual Design Automation Conference, DAC '04, pages 753–760, New York, NY, USA, 2004. ACM. Moderator-Ravi, Srivaths, available at: http://doi.acm.org/10.1145/996566.996771. (Cited on page 2.)

[Lattice corporation 2010a] Lattice corporation. *LatticeXP2 Family Handbook*, February 2010. HB1004, Version 02.5, available at: http://www.latticesemi.com/documents/HB1004.pdf. (Cited on pages 36 and 63.)

[Lattice corporation 2010b] Lattice corporation. *MachX02 Family Handbook*, November 2010. HB1010, Version 01.0, available at: http://www.latticesemi.com/documents/doc39331x10.pdf. (Cited on pages 36 and 63.)

[Lie 2009] Wang Lie and Wu Feng-yan. *Dynamic partial reconfiguration in FPGAs*. In Proceedings of the 3rd international conference on Intelligent information technology application, IITA'09, pages 445–448, Piscataway, NJ, USA, 2009. IEEE Press, available at: http://dl.acm.org/citation.cfm?id=1794754.1794866. (Cited on page 41.)

[MacPherson 1994] Hugh MacPherson. *Tamper respondent enclosure*, 1994. US patent 5,858,500, available at: http://www.patents.com/us-5858500.html. (Cited on page 48.)

[Maghrebi 2011] Houssem Maghrebi, Sylvain Guilley and Jean-Luc Danger. *Leakage squeezing countermeasure against high-order attacks*. In Proceedings of the 5th IFIP WG 11.2 international conference on Information security theory and practice: security and privacy of mobile devices in wireless communication, WISTP'11, pages 208–223, Berlin, Heidelberg, 2011. Springer-Verlag, available at: http://dl.acm.org/citation.cfm?id=2017824.2017844. (Cited on page 80.)

[Menezes 1996] Alfred J. Menezes, Paul C. Van Oorschot and Scott A. Vanstone. *Handbook of Applied Cryptography*, 1996. (Cited on page 6.)

[Messerges 2000] Thomas Messerges. *Using Second-Order Power Analysis to Attack DPA Resistant Software*. In Cryptographic Hardware and Embedded Systems - CHES 2000, volume 1965 of *Lecture Notes in Computer Science*, pages 27–78. Springer Berlin / Heidelberg, 2000, available at: http://dx.doi.org/10.1007/3-540-44499-8_19. (Cited on page 21.)

[Microsemi 2009] Microsemi (ex Actel) corporation. *In-System Programming (ISP) of Actel Low-Power Flash Devices Using FlashPro3*, August 2009. Version 1.5, available at: http://www.actel.com/documents/LPD_ISP_HBs.pdf. (Cited on pages 31 and 33.)

[Microsemi 2010a] Microsemi (ex Actel) corporation. *Actel Fusion Family of Mixed Signal FPGAs*, July 2010. Version 2, rev.1, available at: http://www.actel.com/documents/Fusion_DS.pdf. (Cited on page 63.)

[Microsemi 2010b] Microsemi (ex Actel) corporation. *Fusion FPGA Fabric User Guide*, July 2010. Part Number: 50200265-0, available at: http://www.actel.com/documents/Fusion_UG.pdf. (Cited on pages 29, 35 and 63.)

[Moog] Andreas Moog. *Enterprise Cryptographic Filesystem*, available at: https://launchpad.net/ecryptfs. (Cited on page 57.)

[Quisquater 2001] Jean-Jacques Quisquater and David Samyde. *ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards*. In Isabelle Attali and Thomas Jensen, editeurs, Smart Card Programming and Security, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer Berlin / Heidelberg, 2001, available at: http://dx.doi.org/10.1007/3-540-45418-7_17. (Cited on page 21.)

[Rutkowska 2010] Joanna Rutkowska and Rafal Wojtczuk. *Qubes OS Architecture*, january 2010. Version 0.3, Invisible Things Lab, available at: http://qubes-os.org/files/doc/arch-spec-0.3.pdf. (Cited on page 60.)

[Sang 2010] Fernand Lone Sang, Eric Lacombe, Vincent Nicomette and Yves Deswarte. *Analyse de l'efficacité du service fourni par une IOMMU*. In Symposium sur la Sécurité des Technologies de l'Information et de la Communication, page 189, 2010, available at: http://www.sstic.org/media/SSTIC2010/SSTIC-actes/Analyse_de_l_efficacite_du_service_fourni_par_une_/SSTIC2010-Article-Analyse_de_l_efficacite_du_service_fourni_par_une_IOMMU-lacombe_lone-sang_nicomette_deswarte.pdf. (Cited on page 60.)

[Saout 2004] Christophe Saout. *dm-crypt: a device-mapper crypto target*, 2004, available at: http://www.saout.de/misc/dm-crypt. (Cited on page 57.)

[Shamir 2004] Adi Shamir and Eran Tromer. *Acoustic cryptanalysis On nosy people and noisy machines*, 2004, available at: http://www.cs.tau.ac.il/~tromer/acoustic. (Cited on page 23.)

[Smerdon 2008] Maureen Smerdon. *Security Solutions Using Spartan-3 Generation FPGAs*, April 2008. WP266, Version 1.1, Xilinx corporation, available at: http://www.xilinx.com/support/documentation/white_papers/wp266.pdf. (Cited on page 36.)

[Suh 2003] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk and Srinivas Devadas. *AEGIS: architecture for tamper-evident and tamper-resistant processing.* In Proceedings of the 17th annual international conference on Supercomputing, ICS '03, pages 160–171, New York, NY, USA, 2003. ACM, available at: http://doi.acm.org/10.1145/782814.782838. (Cited on pages 56 and 64.)

[TPM] Trusted Computing Group. *TPM Main Specification.* web site, available at: http://www.trustedcomputinggroup.org/resources/tpm_main_specification. (Cited on page 58.)

[Tzur 2004] Ronen Tzur. *Sandboxie homepage*, 2004, available at: http://www.sandboxie.com. (Cited on page 59.)

[Uhlig 2005] Rich Uhlig, Gil Neiger, Dion Rodgers, Amy L. Santoni, Fernando C. M. Martins, Andrew V. Anderson, Steven M. Bennett, Alain Kagi, Felix H. Leung and Larry Smith. *Intel Virtualization Technology.* Computer, vol. 38, pages 48–56, May 2005, available at: http://portal.acm.org/citation.cfm?id=1069588.1069634. (Cited on page 60.)

[Van Eck 1985] Wim Van Eck. *Electromagnetic radiation from video display units: an eavesdropping risk?* Journal of Computers and Security, vol. 4, pages 269–286, December 1985, available at: http://dl.acm.org/citation.cfm?id=7307.7308. (Cited on page 23.)

[Vaslin 2009] Romain Vaslin, Guy Gogniat, Jean-Philippe Diguet, Eduardo Wanderley, Russell Tessier and Wayne Burleson. *A security approach for off-chip memory in embedded microprocessor systems.* Microprocessors and Microsystems, vol. 33, no. 1, pages 37 – 45, 2009. Selected papers from ReCoSoC 2007 (Reconfigurable Communication-centric Systems-on-Chip), available at: http://www.sciencedirect.com/science/article/B6V0X-4T9CCVR-2/2/f204bac18a3292615e8b1f726beb6e2e. (Cited on page 57.)

[Verayo] Verayo corporation. *PUF: Physically Unclonable Function*, available at: http://www.pufcoinc.com/technology. (Cited on page 41.)

[vsf] (Cited on page 83.)

Vsftpd web page, available at: https://security.appspot.com/vsftpd.html.

[Wikipedia] Wikipedia. *Power analysis*, available at: http://en.wikipedia.org/wiki/Power_analysis. (Cited on page 20.)

[Xilinx] Xilinx corporation. *OpensourceLinux*, available at: http://xilinx.wikidot.com/open-source-linux. (Cited on pages 69 and 83.)

[Xilinx 2009] Xilinx corporation. *Spartan-3AN FPGA Family Data Sheet*, November 2009. DS557, Version 3.2, available at: http://www.xilinx.com/support/documentation/data_sheets/ds557.pdf. (Cited on pages 36 and 63.)

[Xilinx 2010a] Xilinx corporation. *AXI Interconnect IP*, December 2010. DS768 (v1.01.a), available at: http://www.xilinx.com/support/documentation/ip_documentation/ds768_axi_interconnect.pdf. (Cited on page 61.)

[Xilinx 2010b] Xilinx corporation. *LogiCORE IP XPS Central DMA Controller*, July 2010. Product Specification: DS579, Version 2.02a, available at: http://www.xilinx.com/support/documentation/ip_documentation/xps_central_dma.pdf. (Cited on page 70.)

[Xilinx 2010c] Xilinx corporation. *Virtex-6 FPGA Configuration, user guide*, November 2010. UG360 (v3.2), available at: http://www.xilinx.com/support/documentation/user_guides/ug360.pdf. (Cited on pages 42, 52 and 63.)

[Zeineddini 2005] Amir H. Sheikh Zeineddini and Kris Gaj. *Secure Partial Reconfiguration of FPGAs*. pages 155–162, Singapore, December 2005, available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1568540. (Cited on page 43.)

List of Abbreviations

Notation	Description
3-DES	Triple Data Encryption Standard. 29, 37–39
ACS	Advanced Crypto-Signer. 86
AES	Advanced Encryption Standard. 29, 31, 37, 38, 43, 44, 48, 49, 52, 57, 105
AES-CBC	AES in CBC mode. 49, 50
AES-GCM	AES in GCM mode. 44, 79
AMBA	Advanced Microcontroller Bus Architecture. 49
AMD	Advanced Micro Devices. 59, 60, 105
AMD-V	AMD virtualization. 59
ANSSI	Agence nationale de la sécurité des systemes d'information. 11
ARM	Advanced RISC Machines. 61
ASIC	Application-Specific Integrated Circuit. 1, 14, 15, 19, 64, 65
AXI	Advanced eXtensible Interface. 49, 50, 76, 85
BiT	Board is Trusted. 42, 44, 47–50, 52
BRAM	Block RAMs. 47, 49, 50, 65–69
CBC	Cipher Block Chaining. 49, 105
CC	Common Criteria. 8, 11
CCTL	Common Criteria Testing Laboratory. 11
CEMA	Correlation ElectroMagnetic Analysis. 22
CESTI	Centre d'Evaluation de la Sécurité des Technologies de l'Information. 11
CF	Compact Flash. 49, 50
CL	Configuration Logic. 16, 43, 52
CLB	Configurable Logic Block. 1
CMOS	Complementary Metal Oxide Semiconductor. 17
Cofrac	Comité français d'accréditation. 11
CRC	Cyclic Redundancy Check. 29, 57
CTCPEC	Canadian Trusted Computer Product Evaluation Criteria. 8
DDR	Double Data Rate. 50, 79, 105
DDR3	DDR 3rd generation. 50
DEMA	Differential ElectroMagnetic Analysis. 22, 77

Notation	Description
DFA	Differential Fault Analysis. 77
DMA	Direct Memory Access. 60, 70, 71
DMAR	Direct Memory Access Remapping. 60
DoS	Denial of Service. 28, 35, 42
DPA	Differential Power Analysis. 20–22, 77
DPR	Dynamic Partial Reconfiguration. 3, 27, 41, 43, 44, 52, 88, 91, 107
DSP	Digital Signal Processor. 16, 50
EAL	Evaluation Assurance Level. 8, 9
ECS	Embedded Crypto-Signer. 83, 88
EEPROM	Electrically Erasable Programmable Read-Only Memory. 18
EMA	ElectroMagnetic Analysis. 21, 22
EPROM	Erasable Programmable Read-Only Memory. 18
FIPS	Federal Information Processing Standard. 11
FiT	FPGA is Trusted. 42, 44, 47–50, 52
FPGA	Field-Programmable Gate Array. 1–3, 5, 14–19, 25–29, 31–44, 47–49, 52, 55–69, 73, 75, 76, 88, 91, 92, 105, 111
FSM	Finite State Machine. 32, 38, 39
FTP	File Transfer Protocol. 83, 107
GCM	Galois Counter Mode. 44, 57, 105
GPP	General Purpose Processor. 77, 80, 86
GPS	Global Positioning System. 1
HDL	Hardware Description Language. 15, 16, 76
HMAC	Hash-based MAC. 43, 44, 47–50, 52, 105, 106
HMAC-SHA1	HMAC-SHA1. 43
HMAC-SHA256	HMAC-SHA256. 50
HoD	Hardware on Demand. 41
HO-DPA	High-Order Differential Power Analysis. 21
HWICAP	HardWare ICAP. 50
IBM	International Business Machines. 7, 61
IC	Integrated Circuit. 14, 48
ICAP	Internal Configuration Access Port. 16, 50, 106
IOMMU	Input / Output Memory Management Unit. 60, 61
IP	Intellectual Property. 16, 23, 28, 31, 58, 70, 85

Notation	Description
ISP	In-System Programming. 31, 35
ITSEC	Information Technology Security Evaluation Criteria. 8
JTAG	Joint Test Action Group. 16, 37, 38
LUT	Look Up Table. 1, 16
MAC	Message Authentication Code. 31, 34–36, 43, 105
MI5	Military Intelligence, Section 5. 22
MMU	Memory Management Unit. 59, 60
NIST	National Institute of Standards and Technology. 11, 68
Nonce	Number used once. 48
NSA	National Security Agency. 11, 23
NVM	Non-Volatile-Memory. 28, 31, 33, 36, 57, 62, 69
OS	Operating System. 3, 55, 56, 59, 61, 62, 64, 69, 76, 88, 91
OTP	One-Time Pad or One-Time Password. 57
PAR	Place And Route. 49
PIN	Personal Identification Number. 83
PLB	Processor Local Bus. 70, 83, 85
PLL	Phase-Locked Loop. 85
PP	Protection profile. 8
PRNG	Pseudo-Random Number Generator. 85
PRV	Protected Random Value. 56
PUF	Physically Unclonable Function. 41
RAM	Random Access Memory. 15, 16, 36, 37, 44, 47, 50, 56, 57, 61, 65, 66, 68, 70, 71, 105
RCS	Reconfigurable Crypto-Signer. 88
RISC	Reduced Instruction Set Computing. 61, 105
RM	Reconfigurable Module. 16, 41
RoT	Root of Trust. 44
RP	Reconfigurable Partition. 16, 41, 46, 47
RSA	Rivest, Shamir and Adleman. 20, 68, 70
RSA-PSS	Rivest, Shamir and Adleman - Probabilistic Signature Scheme. 64

Notation	Description
SD	System Designer. 16, 19, 29, 32–36, 38–40, 43, 44, 47, 66, 67
SDK	Software Development Kit. 49
SecURe DPR	Secure Update against Replay attacks for DPR. 3, 27, 44, 47–49, 52
SEMA	Simple ElectroMagnetic Analysis. 22, 77
SFR	Security Functional Requirement. 8
SHA	Secure Hash Algorithm. 43, 49, 50, 68–71, 88, 105, 106
SiP	System in Package. 73
SPA	Simple Power Analysis. 20–22
SRAM	Static Random Access Memory. 3, 17, 18, 23, 26–28, 41, 43, 52, 68, 69, 91
ST	Security Target. 8
STRES	Secure Techniques for Remote reconfiguration of Embedded Systems. 31
STS	Station-to-Station. 31
System ACE	System Advanced Configuration Environment. 50
TCG	Trusted Computing Group. 57
TCP/IP	Transmission Control Protocol / Internet Protocol. 31
TCSEC	Trusted Computer System Evaluation Criteria. 8
TEMA	Template ElectroMagnetic Analysis. 22
TOE	Target Of Evaluation. 8
TPM	Trusted Platform Module. 57, 58
TRNG	True Random Number Generator. 85
UART	Universal Asynchronous Receiver Transmitter. 44, 50
UL	User Logic. 16, 43, 44, 47, 48
UV	UltraViolet. 18
VHDL	VHSIC Hardware Description Language. 15
VHSIC	Very-High-Speed Integrated Circuit. 15, 107
VM	Virtual Machine. 59, 60
vsFTPd	Very Secure FTP Daemon. 83
VT-d	Virtualization Technology for Directed I/O. 60
VT-x	Virtualization Technology. 59
XPS	Xilinx Platform Studio. 49, 70

I'll be back.

Terminator T-800 model 101, Terminator

Securing embedded systems based on FPGA technologies

Abstract: Embedded systems may contain sensitive data. They are usually exchanged in plaintext between the system on chips and the memory, but also internally. This is a weakness: an attacker can spy this exchange and retrieve information or insert malicious code. The aim of the thesis is to provide a dedicated and suitable solution for these problems by considering the entire lifecycle of the embedded system (boot, updates and execution) and all the data (FPGA bitstream, operating system kernel, critical data and code). Furthermore, it is necessary to optimize the performance of hardware security mechanisms introduced to match the expectations of embedded systems. This thesis is distinguished by offering innovative and suitable solutions for the world of FPGAs.

Keywords: FPGA, Reconfigurable architecture, Bitstream, Replay attack, Boot, Embedded operating system, Dynamic partial reconfiguration

Sécurisation des systèmes embarqués basés sur les technologies FPGA

Résumé: Les systèmes embarqués peuvent contenir des données sensibles. Elles sont généralement échangées en clair entre le système sur puces et la mémoire, mais aussi en interne. Cela constitue un point faible: un attaquant peut observer cet échange et récupérer des informations ou insérer du code malveillant. L'objectif de la thèse est de fournir une solution dédiée et adaptée à ces problèmes en considérant l'intégralité de la durée de vie du système embarqué (démarrage, mises à jour et exécution) et l'intégralité des données (bitstream du FPGA, noyau du système d'exploitation, code et données critiques). En outre, il est nécessaire d'optimiser les performances des mécanismes matériels de sécurité introduits afin de correspondre aux attentes des systèmes embarqués. Cette thèse se distingue en proposant des solutions innovantes et adaptées au monde des FPGAs.

Mots clés: FPGA, Architecture reconfigurable, Bitstream, Attaque par rejeu, Boot, Système d'exploitation embarqué, Reconfiguration dynamique partielle