Selected Challenges on Security and Dependability of Embedded Systems

Przemysław Osocha[1], João Carlos Cunha[2], and Francesca Matarese[3]

[1,3]SESM s.c.a.r.l., Naples, Italy
[1]p.osocha@gmail.com, [3]fmatarese@sesm.it
[2]Polytechnic Institute of Coimbra / CISUC, Coimbra, Portugal
jcunha@isec.pt

Abstract. As embedded systems are silently spreading into every corner of our technological society and being interconnected in a world-wide network, they become invaluable to our daily lives, therefore our concern in their correct and secure behavior increases. However, improving dependability and security of embedded devices must be done with special care, since they are usually developed under severe resource- and price-constraints.
In the frame of Critical Step project some FPGA-based embedded systems have been developed, with special concern for dependability and security improvement. Due to resource constraints of these devices, some particular challenges were faced, that are described in this paper: when to employ security during system lifecycle; which security threats to deal with; how to cope with computational-demanding cryptography; how to deal with security in safety-critical systems; how to increase dependability; how to assure timeliness; and how to evaluate dependability.

Keywords: security, dependability, embedded system, FPGA, partial reconfiguration

1 Introduction

Embedded systems are becoming available anytime and anywhere in many different forms. They may be evident and visible or hidden and integrated into equipment, devices and environment. They are: mobile devices like smartphones; household appliances like HVAC (heating, ventilation, and air conditioning); automatic equipment in the streets like traffic controls; complex airport systems like the ones for surveillance. They are seamlessly and pervasively blending into modern human environment. The constant development in embedded systems industry drives these devices to be incorporated into every product, replacing manual or mechanical solutions. The embedded electronics components share in the final products is constantly increasing, especially in ICT (information and communication technologies) and health domains.

This irrepressible spreading of embedded systems is nowadays accompanied by a new phenomenon, that is connectivity. All those small devices, often integrated into bigger products, have at their disposal wide range of possibilities to connect and ex-

change data. They are connecting through wired or wireless networks, utilizing numerous communication standards. This situation poses absolutely new threats for security of systems. Dependability and security cannot be considered anymore an aspect of single, separate device, but rather depending on application field, as dependability and security of system of embedded systems.

Taking advantage of researchers' mobility provided by Critical Step project [7], some research and development projects have been initiated, with the aim of exploring security and safety of embedded systems. One of these projects consisted in the definition of an architecture framework for a networked embedded node with intrinsic security, privacy and dependability capabilities [8]. An other project consisted in the development of a secure and dependable multilateration navigation system (MLAT), which provides high-performance and accurate all-weather surveillance, supporting full control of the airport surface and near-airport airspace [10].

Common to these projects was the fact that they utilized FPGA-based development boards, namely a Xilinx FPGA Virtex-5 ML507 Evaluation Platform [12], and an Altera FPGA Cyclone III 3C120 Development Board [13].

During the development of these systems, some problems that challenged the traditional approach to security and dependability, as applied to ICT systems, were faced, due to intrinsic characteristics of embedded systems, such as resource constraints and long life expectancy. These challenges are analyzed in the following sections. After defining, in Section 2, the scope of the studied embedded systems, in Section 3 some security challenges are presented, such as the need to employ security mechanisms both during the system development and its operational life, the identification of security threats, the management of computational-demanding cryptography schemes, and particularities of safety-critical systems. Then, in Section 4, dependability challenges are addressed, such as how to manage space redundancy for providing system dependability, how to guarantee timeliness in fault-tolerant systems, and how to evaluate dependability in such systems. The paper concludes with summary of considered challenges.

2 Embedded Systems

Embedded systems are computer devices designed for specific functions that are incorporated into enclosed products with hardware and mechanical parts, often with real-time constraints [1]. Although this definition is generic enough, it does not reveal the huge differences that exist between embedded devices, related to capabilities, architecture or technology.

2.1 Embedded Systems Complexity

The diversity in capability requirements for embedded devices is so large that a unified architectural or technological approach is impracticable. A wireless sensor node, for example, has very simple hardware and software requirements, being very sensitive to power consumption. On the other hand, an embedded camera array with image

processing requires high performance computing, large memory storage capacity, and high-speed network connection.

These different requirements lead to large variety in the assortment of technologies and architectures. Many of current low-performance embedded devices are deployed with 4- or 8-bit microcontrollers, running simple single-processing applications, or even executing discrete logic without any processor or software. These systems are very cost-sensitive, since millions of units can be built, thus functionality is often the major concern, in detriment of dependability or security.

On the other side of complexity spectrum are systems using multiple pipelined microprocessors, running real-time operating systems. If deployed in high quantities, these complex systems are usually build on top of devices with microprocessors; otherwise it could be economically advantageous to use FPGA-based systems with synthesized microprocessors (see Section 2.2).

In-between, there are systems that don't have any processing unit, but are high-demanding. They use discrete-logic for performing specialized operations, such as signal processing or video encoding, and are deployed on FPGAs or, if large quantities are required, in ASICs (application-specific integrated circuit) because their prices are much lower.

2.2 FPGA-based Embedded Systems

An FPGA (field-programmable gate array) is an integrated circuit that can be easily reprogrammed during prototyping or production, giving the engineers the possibility of using different architectural hardware solutions depending on the requirements and complexity of the system. It is possible to combine different discrete-logic cores, such as video controllers, Ethernet controllers, data encryption/decryption modules, etc. and even microcontroller or microprocessors. If a microcontroller or microprocessor is present, it is possible to run on it an operating system and/or application software, turning this device into a system on a chip (SoC).

However, what is becoming really interesting, is that FPGA-based systems can be also reprogrammed after deployment, and thus it is possible to correct design errors or even upgrade it to adopt the new functionalities. The most popular FPGA devices are based on SRAM (Static Random Access Memory). They are composed of a matrix of configurable logic blocks, interconnected with routing elements, stored in a configurable memory. Every time the device is powered on, this configuration memory is programmed with a bitstream that contains the logic information describing the device. This bitstream is copied from a flash memory, usually by a microcontroller.

2.3 Characteristics of the Studied Devices

The devices studied and developed within the scope of Critical Step project were high-demanding, both in terms of computing power and processing complexity. Even if they were considered in the frame of a research project, it was decided to adopt commercial and well known FPGAs as means for further development of embedded systems, namely a Xilinx Virtex-5, and an Altera Cyclone III. These FPGAs have

been successfully used with discrete logic, with soft-cores and hard-cores, and software applications written in C language.

3 Security Challenges

Many embedded systems currently available on the market completely lack of security, leaving them vulnerable to attacks. One of the main reasons for this weakness is the struggle for company's competitiveness on the market. Products have to hit the market as soon as possible, with the lowest possible cost, thus limiting physical resources to the minimum. Only key functionalities are implemented, and frequently security is not considered as one of them, but rather as an added cost with no corresponding value.

3.1 Security in System Lifecycle

Updating an embedded system after deployment is usually difficult or even impossible to implement. Systems are dimensioned with only the necessary resources, thus any change is often impossible to accommodate, or implies a thorough redesign. Therefore security usually cannot be just added later on, it must be planned and designed during development, so it can be assigned the necessary physical and time resources.

However, security provision doesn't end with system deployment; it should be addressed like a process that needs to be maintained throughout the whole product lifetime. During development it is possible to test and assure product immunity against currently existing treats, but in time new treats will appear and security can be impaired by discovery of new ways of attack. For example, cryptographic algorithms can be well accepted and considered secure in time of product design, but then are broken due to new discovered weaknesses or just by brute force attacks of more and more powerful computing units. The security of the product has to be continuously monitored and updated if necessary, to assure the required level of system security protection against attacks. Periodical evaluation of product security against new exploits and attacks should be part of the product lifecycle maintenance. It is important to remember, that embedded devices often have a much longer life than ICT systems.

One solution used by industry for long life installations is the utilization of FPGA-based embedded devices. The usage of FPGAs allows the development and installation of hardware one time only, and then, if necessary, the software and programmable part of the FPGA logic are simply upgraded. In particular, that approach is used in design of airports radar systems, where FPGA-based embedded devices operate for years, but hardware adjustments according to changes in standards are still possible thanks to the ability of the FPGA chip to reprogram its logic.

The first identified challenge consists in providing security capabilities during system development, and also reserving enough space to accommodate for latter improvements that eventually will be necessary. This means that, for FPGA-developed systems, the interfaces between the security units and the rest of the system must be

carefully designed, extra space must be reserved, and time constraints must be calculated to allow for spare time that can accommodate future security enhancements.

3.2 Security Threats Identification

The attacker may gain access to the system in basically two ways [4]: physically, or remotely. If the embedded device is physically accessible to the attacker, then he has almost unlimited possibilities for tampering the product, by opening the chassis, understanding hardware design, recognition of used chips, extraction of firmware code, reverse engineering, etc. This type of attacks is particular to embedded devices, and has been studied for long time.

However, the most important security threat nowadays comes from intruders that attack remotely, have no need for special tools and equipment, and can easily hide their location by masking their own connection to the internet. As embedded devices become networked, they get vulnerable to remote attacks.

Although this is a problem that affects all systems connected to the internet, most methods used in ICT systems do not easily apply to embedded system devices. For example, due to limited resources, it is usually not possible to have strong cryptographic algorithms running; and due to a closed design it is not easy to apply patches with security updates.

Furthermore, these systems are more vulnerable to certain attacks than other ICT systems [5]: since they usually interact with the physical environment, they must act in real-time. A continuous attack on an embedded device may interfere with system timing, introducing delays that can cause missing deadlines, and the loss of control-loop stability. This kind of attacks can also provoke higher power consumption, which could drain the device's battery.

The challenge is thus to identify the possible security threats to which the embedded system can be exposed, and then address them in a proper way, by providing special capabilities to the devices. For example, a low-voltage detection circuit can physically disconnect the network access, or a tampering detection mechanism can trigger the deletion of sensitive data stored in the device, safeguarding confidentiality.

3.3 Security and Cryptography

Strong cryptographic algorithms have high computational demands. They are usually built on software, requiring a microprocessor to be able to run. When the embedded device is based on FPGA, a soft- or hard-core microprocessor must be provided. Since synthesized models of microprocessors on FPGA run slowly, encrypting/decrypting may take a long and unpredictable time. The solution can be the choice of a simpler algorithm, with the disadvantage of being easier to break.

Another possible approach is to substitute software running on top of a microprocessor by a synthesized model of the cryptographic algorithm deployed in the FPGA. This assures that the same complex algorithm may execute much faster in a system with lower clock speed and lower power consumption. In fact, due to parallelization and low-level computation, it runs even faster than on a real microprocessor.

Some advices can be considered by developers. Elliptic curve cryptography (ECC) is an approach to public-key cryptography well accepted in embedded systems world thanks to its limited resource requirements [9]. Worth to mention is also eSTREAM project, that aims providing modern, efficient and compact stream ciphers, designed specifically for software or hardware implementations [11].

The capability of an FPGA to be reconfigured during its lifetime gives the possibility to upgrade the cryptographic scheme. For example, one of the best measures to have a password undiscovered is to change it frequently. A cryptographic key, or even the encryption/decryption algorithm itself, can be changed periodically, decreasing the chances of a successful attack.

The challenge is thus, depending on application field, to find a strong cryptographic scheme, able to process in a predictable time-bound in order to have real-time guarantee, and have it available for an FPGA. The FPGA-based device should be ready for secure upgrade, to update the cryptographic key or algorithm in case of key compromising or cipher weaknesses discovery.

3.4 Safety-critical Systems

Security in embedded systems often implies safety issues [3], as opposite to most ICT systems. For example, if an encryption algorithm key is compromised and a malicious action is issued against a financial institution, it usually doesn't involve the loss of lives or any major environmental disaster. Furthermore, it might be possible to revert the effects of this attack. If an embedded controller is affected by a malicious attack, for example, against a train control and signaling systems, causing a railroad accident, it is not possible to *roll-back*.

When designing dependable embedded devices, the designers usually take into consideration a fault model based on physical interference, where faults have an external non-malicious origin, thus affecting the device's components in a randomly distributed way in space and time, and provoking stuck-at or bit-flip faults. When considering malicious and deliberate faults, this random and bit-flip distribution of hardware faults is no longer valid, as attackers may try, for example, to repeatedly inject faults or interfere with data communications.

The challenge here is to develop a new fault model, taking into account faults due to possible malicious remote access to device, and possible application as a part of bigger safety-critical system.

4 Dependability Challenges

Dependability is achieved through redundancy, in a combination of space, time and information. Embedded systems, having stringent resource constraints, oblige a careful utilization of space (either redundant hardware, or software or information) and, due to real-time performance constraints, also demand for a cautious utilization of time redundancy (due to repeated computation or redundant software).

In this section only transient hardware faults that may affect the FPGA configuration memory are addressed. These faults are much more common than permanent or intermittent faults. Software faults are not addressed, since the focus of this paper is mainly on FPGA related dependability issues.

4.1 Hardware Redundancy Reduction

Replication of hardware modules is a common way of implementing hardware fault tolerance. It may take the form of static redundancy, such as triple modular redundancy (TMR), where replicated modules produce results in parallel, that are voted or compared; or dynamic redundancy, such as standby spare, where the redundant modules are in standby until they are necessary to replace a failed active module. If these spare modules are executing in parallel with the active module, although their outputs are ignored, it is said they are in hot standby; if they are not executing, they are in cold standby.

The capability of an FPGA to be reprogrammed during its operational life, opens the possibility of having the spare modules in cold standby outside of the FPGA, i.e., of reprogramming the FPGA with a spare module only when it is necessary, either in a different part of the FPGA, or even substituting the failed module. This last option may simply result from a restart of the system, because during power-on the FPGA is always reprogrammed.

Concerning static redundancy, or dynamic redundancy in hot standby, it is obvious that it is not possible to avoid the existence of the replicated modules, since they must execute in parallel; however, it is possible to recover a failed replica or former active module, by reprogramming it, and thus a system with degraded fault-tolerance can fully recover its reliability (the failed replica becomes an active redundant module, or the former active module becomes a new spare module in active standby).

The main issue regarding FPGA reconfiguration relates to the introduced time overhead, which may compromise system reliability, since reconfiguration takes time. Section 4.2 will address this challenge.

The current challenge is thus to choose an adequate fault-tolerant model, considering the limitations in terms of available space and time.

4.2 Time Challenge in System Recovery

Stopping a system for repair is usually not possible in embedded systems, mainly if they are operating in real-time. For example, in a control system, if the controller stops for repair, it will probably lose control over the process.

Having static redundancy or replicas in hot standby allows the continuous operation of the system when an active module fails. However, in order to repair this module it is necessary to reprogram the FPGA. With spare modules outside the FPGA, the issue is the same. Traditional FPGA reprogramming implies that it must stop for transferring the bitstream with logical information into the SRAM configurable memory.

Some FPGA manufacturers, namely Xilinx, offer now the possibility of dynamic partial reconfiguration, meaning that it is feasible to reprogram only a part of the FPGA, while the remaining part operates uninterruptedly [2]. This opens the possibility to repair the failed replicas, while the system runs in degraded mode, but without losing control over the controlled process.

This is, however, not a solution for replacing an active module by a replica in cold standby, as it always takes time to reprogram, even partially, the FPGA. This situation may be addressed by taking advantage of the dynamics of the controlled process. Since every physical process has some inertia, it is usually possible to remain without control for some time [6]. The question is if reprogramming the FPGA is fast and predictable enough for the control system to tolerate this gap in the control-loop.

This challenge is related to the previous one: to choose an adequate fault-tolerant model for a particular application, considering space and time limitations.

4.3 Fault-injection for Dependability Evaluation

Fault-injection is a well-known methodology for dependability evaluation of computer systems. Faults may be injected by introducing external disturbances in power supply or chip pins, or directly inside the chips by using heavy-ion radiation or laser. However, with these techniques, experiments are difficult to control and to repeat. Software implemented fault-injection (SWIFI) is, though, very popular and allows high controllability of experiments. The drawback is that software must run inside the target embedded system, which naturally interferes both with memory space and processing time.

Regarding the injection of faults into an FPGA, other options exist, such as:

1. Using a fault-injector module programmed inside the FPGA itself, that may interfere with signals, memory cells, etc.
2. Reprogramming the FPGA (partially and dynamically) with an already corrupted bitstream emulating the fault.
3. Using boundary-scan to corrupt the configuration memory (the use of boundary-scan is a technique already widespread to inject faults in functional units of a system. It is known as Scan-Chain Implemented Fault Injection – SCIFI).

The first of the above described options have been used, by introducing a small module able to interfere in a predefined way with a specific signal in the device. This is a simple way to inject a fault, but with a limited scope, useful only for particular situations. Therefore more general techniques must be studied, posing a demanding challenge.

5 Conclusion

The mass deployment of networked embedded systems with critical control functions requires concentrating, more than ever, on security and dependability matters. This poses many new challenges to designers of such systems. In frame of this work, vari-

ous embedded systems based on FPGA devices were developed. During this research numerous challenges, related to dependability and security implementation in resource-constrained systems, were faced.

In this paper some of those challenges were presented and discussed, together with proposals on possible solutions. In case of security of FPGA-based embedded systems, some possible solutions of the following challenges were proposed: the necessity to implement security mechanisms during design phase and maintain it throughout the whole system life; the identification of security threats; the selection of appropriate cryptography schemes; and security of safety-critical embedded systems. In case of dependability, the paper introduced the following challenges: the management of space redundancy; the issues of timeliness assurance in fault-tolerant systems; and evaluation of dependability in such systems.

It must be stressed that this list does not exhaust all the embedded systems issues related to dependability and security; it only contains those considered the most challenging during Critical Step project. Apart from the proposed possible solutions for the presented challenges, the main goal of the paper was to alert and justify their importance for future development of embedded systems.

6 References

1. Marwedel, P.: Embedded System Design (Embedded Systems Foundations of Cyber-Physical Systems). TU Dortmund, Informatik. Dordrecht: Springer Netherlands (2011)
2. Rana, V., Santambrogio, M., and Sciuto, D.: Dynamic Reconfigurability in Embedded System Design. In: IEEE International Symposium on Circuits and Systems ISCAS'2007. New Orleans (2007)
3. Schoitsch, E.: Design for Safety and Security of Complex Embedded Systems: A Unified Approach. In: Kowalik J.S., Gorski J., Sachenko A. (eds.), Cyberspace Security and Defense: Research Issues, vol. 196 of NATO Science Series II - Mathematics, Physics and Chemistry, pp. 161-174 (2005)
4. Grand, J.: Practical Secure Hardware Design for Embedded Systems. In: Proceedings of the 2004 Embedded Systems Conference (2004)
5. Koopman, P.: Embedded System Security. IEEE Computer, vol. 37, no. 7, pp. 95–97, July (2004)
6. Cunha, J.C., Maia, R., Rela, M.Z., Silva, J.G.: A Study on Failure Models in Feedback Control Systems. In: International Conference on Dependable Systems and Networks, Goteborg, Sweden (2001)
7. CRITICAL STEP project: The CRITICAL Software Technology for an Evolutionary Partnership. A Marie-Curie Industry-Academia Partnerships and Pathways (IAPP) project belonging to call FP7-PEOPLE-2008-IAPP, website: `http://www.critical-step.eu/`
8. pSHIELD project co-funded by the ARTEMIS Joint Undertaking. Research of Security, Privacy and Dependability in context of Embedded Systems, website: `http://www.pshield.eu/`
9. pSHIELD project deliverables D2.3.2, D3.3, D6.2, D6.3, accessed at `http://pshield.unik.no/wiki/PublicDeliverables`
10. ADAM/AWAM Systems: Multilateration Solutions by SELEX Sistemi Integrati, `http://www.selex-`

si.com/IT/Common/files/SelexSI/brochure_datasheet/2008/Data_Sheet/Adam.pdf
11. eSTREAM project. The ECRYPT Stream Cipher Project, website: http://www.ecrypt.eu.org/stream/
12. Xilinx: ML505/ML506/ML507 Evaluation Platform – User Guide. UG347 (v3.1.2) May 16 (2011)
13. Altera: Cyclone III 3C120 Development Board – Reference Manual. Document version 1.2, March (2009)