Microprocessors and Microsystems 42 (2016) 127–141

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Protection of heterogeneous architectures on FPGAs: An approach based on hardware firewalls

Pascal Cotret[a,*], Guy Gogniat[b], Martha Johanna Sepúlveda Flórez[b,c]

[a] IETR/SCEE, CentraleSupélec, Avenue de la Boulaie, CS 47601, F-35576 Cesson-Sévigné cedex, France
[b] Lab-STICC Laboratory, University of South Brittany, Lorient, France
[c] Institute for Security in Information Technology, Technical University of Munich, Germany

ARTICLE INFO

Article history:
Received 24 October 2014
Revised 20 June 2015
Accepted 20 January 2016
Available online 6 February 2016

Keywords:
Security
Hardware
MPSoC
Communications
FPGAs
Embedded systems

ABSTRACT

Embedded systems are parts of our daily life and used in many fields. They can be found in smartphones or in modern cars including GPS, light/rain sensors and other electronic assistance mechanisms. These systems may handle sensitive data (such as credit card numbers, critical information about the host system and so on) which must be protected against external attacks as these data may be transmitted through a communication link where attackers can connect to extract sensitive information or inject malicious code within the system. This work presents an approach to protect communications in multiprocessor architectures. This approach is based on hardware security enhancements acting as firewalls. These firewalls filter all data going through the system communication bus and an additional flexible cryptographic block aims to protect external memory from attacks. Benefits of our approach are demonstrated using a case study and some custom software applications implemented in a Field-Programmable Gate Array (FPGA). Firewalls implemented in the target architecture allow getting a low-latency security layer with flexible cryptographic features. To illustrate the benefit of such a solution, implementations are discussed for different MPSoCs implemented on Xilinx Virtex-6 FPGAs. Results demonstrate a reduction up to 33% in terms of latency overhead compared to existing efforts.

1. Introduction

For many years, embedded systems are used in our daily life: we found them in electronic devices, automotive applications, telecommunications systems and so on. When designing such systems, several issues have to be taken into account and one of the major concerns is about security. Since the late 90s, security has become a key point in the development of embedded systems [1]. The number of weaknesses is in constant progress and electronic devices have to process data with various security requirements. According to [2], security criteria are communications security, storage security, inputs/outputs security and users authentication. This work focuses on the two first criteria (communications and storage).

First of all, this work considers communication protection as a key point in embedded systems development as communications channels convey several data types (application codes, confidential data, cryptographic elements and so on) with various needs in terms of security: confidential data must not be revealed to an unauthenticated user while application may be accessible through a specific interface (for instance, for development purposes). Then, this work also takes care of data storage security: memory elements are another critical entry point for attackers as they potentially contain plaintext data.

This work is organized as follows. Section 2 presents related works and our constraints regarding the architecture. Sections 3 and 4 describe our solution in a static and dynamic approach. Section 5 gives an analysis in terms of security and provides implementation results in comparison with other approaches.

2. Scientific context

2.1. Related works

Several studies dealing with security in embedded systems have been published [1,2]. Security mechanisms can be implemented in two ways: hardware blocks or software functions. Software solutions are generally slower, in terms of latency, than a pure hardware-implemented security solution. Furthermore

* Corresponding author. Tel.: +33 299844577.
E-mail addresses: pascal.cotret@centralesupelec.fr, pascal.cotret@supelec.fr (P. Cotret), guy.gogniat@univ-ubs.fr (G. Gogniat), johanna.sepulveda@tum.de (M.J. Sepúlveda Flórez).

http://dx.doi.org/10.1016/j.micpro.2016.01.013
0141-9331/© 2016 Elsevier B.V. All rights reserved.

software solutions are generally more easily compromised than hardware countermeasures. In this section, several works about memory protection are presented. Then, regarding internal transactions protection, an overview of the main solutions is proposed.

2.1.1. Memory protection approaches

In order to provide countermeasures against the threat model defined in Section 2.2, a key point is to address memory protection. An obvious solution is to implement cryptographic features for memory confidentiality and integrity. XOM [3] is a solution mixing confidentiality and integrity for systems where the external memory can be tampered. The implementation requires adding hardware modules and modifying the processor structure. Using such a solution, performances are quite spoiled as authors [3] announce a 50% loss. AEGIS architecture [4] is another approach based on a security-enhanced processor embedding confidentiality and integrity functions. Depending on processor configuration (cache size), memory slowdown is between 3.8% and 130%. Bossuet et al. [5] made an in-depth comparison of existing cryptographic processors where some of them, such as HCrypt, were implemented on FPGAs. Some of these processors are efficient but do not cover our threat model defined in a further section. In [6] authors describe a solution (called SecSoft) to protect software with a hardware Encryption Management Unit (EMU). This work proposes a latency analysis of several modes (block/counter modes for encryption function and with/without encryption). Latency overhead on a ML301 platform goes from less than 10% (block mode, unencrypted) up to 80% (block mode, encrypted). This solution does not provide mechanisms targeting integrity. Elbaz et al. [7] proposes the PE-ICE solution to check integrity in parallel to encryption (i.e. confidentiality). The worst case implementation shows a performance loss of 20% and for a confidentiality only implementation, a 4% loss is given. Vaslin et al. [8] proposes a confidentiality and integrity hardware block based on AES for confidentiality and cyclic redundancy check for integrity, performance loss is about 13–14%. In [9], authors extended this work by using the AES-GCM algorithm (this option is also used in this work), integrity is done by a low latency function [10]. Other methods such as hash trees and formal verification [11,12] are used to protect memory contents.

Another solution is to use the built-in MMU (Memory Management Unit) available in some processors. This work is implemented on a Xilinx FPGA where the softcore Microblaze is provided as a general purpose processor. Microblaze MMU [13] provides a simple access control allowing to get read-only or full-access memory pages in the system, this control can be disabled in a configuration register.

All these works propose solutions to provide encryption methods for external memories protection with different performance versus security tradeoff. In order to protect the target system from attacks on the external memory, a trivial solution consists in building a fully-protected external memory unit. Unfortunately, in this case, each memory access implies a ciphering/deciphering latency penalty. Thus, such an approach is strongly penalizing regarding the overall latency overhead of an application. To mitigate this point, our work proposes to implement cryptographic features only on specific memory pages, defined by application requirements, avoiding such a systematic latency penalty. Therefore, some pages are still not protected, that is the reason why internal traffic protection and/or monitoring must be also addressed in the context of embedded systems security.

2.1.2. Bus and NoC-based security methods

Regarding large scale systems with NoC-based communication architecture, Diguet et al. [14] proposes a solution where security controls are done in each network interface in a distributed manner. In this case, a security manager unit gathers individual interfaces information and performs countermeasures and security updates (done through dynamic partial reconfiguration). This method takes into account processor facing denial of service attacks but does not offer ciphering function (however, authentication and integrity features are available). Fiorin et al. [15–17] propose an alternative to this approach providing “security sensors” inside network interfaces to refine controls (NI in Fig. 1). These sensors are able to block incoming malicious data when parameters are not proven. These parameters are stored in a trusted CAM (Context-Addressable Memory). Finally, a SNM (Security Network Manager) gathers information from NIs to detect potential collisions and errors in data traffic. Unlike [14], security mechanisms can be updated without partial reconfiguration of the FPGA chip, update is done using memory rewriting. Refs. [14] and [15–17] do not offer cryptographic features to cipher data transmitted in the communication network.

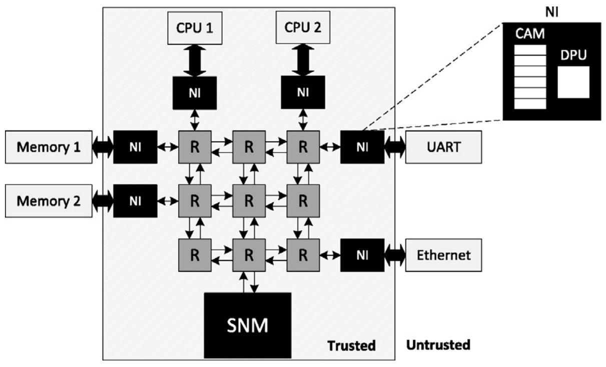

Fig. 1. Fiorin's approach [15,17].

The solution proposed by Fiorin et al. is based on a secured Network Interface (NI) with a DPU mechanism (Data Protection Unit, see Fig. 1). This mechanism allows or not a transaction according to parameters stored in individual trusted CAM memories. [15] requirements aim to cover a threat model with denial of service attacks but cannot protect systems against IP modifications performed by an attacker. This distributed approach has a low-latency and also presents some robustness. Controls are done in each interface, even if one of them is corrupted, other interfaces should still continue to work. Regarding NoC-based security solutions, we can also cite [18,19]. Structure proposed by Sepulveda et al. is based on a hierarchical NoC with low NoCs (each low NoC is a sub-network having a single security policy) and a single high NoC acting as a global security manager (connections with each low NoC). Regarding NoC-based security, LeMay et al. [20] propose a mechanism to detect abnormal behaviors in a NoC protocol when a malicious IP is inserted in the system architecture: the solution based on AXI signals has an area overhead up to 23%.

Then, for bus-based MPSoCs, the main contribution was published by Coburn et al. [21]. This approach is similar to [15–17] as it is based on security-enhanced network interfaces called SEI (*Security Enforcement Interface*) but in a centralized approach. The main drawback of this solution is that security information is sent to a global security manager which is the only component able to perform controls. Therefore, latency overhead is increased (see Section 5). This solution does not offer security updates or cryptographic features. Coburn et al. approach suggests to centralize all the controls in a single module; therefore, as soon as this module is corrupted, system security is compromised.

Compared to these efforts, our approach provides a distributed solution with update mechanisms. We are able to dynamically adapt the security policies based on the instantaneous threats. We