American Journal of Embedded Systems and Applications
2014; 2(1): 1-5
Published online February 28, 2014 (http://www.sciencepublishinggroup.com/j/ajesa)
doi: 10.11648/j.ajesa.20140201.11

Enhance performance in implementing the security of partially reconfigurable embedded systems

Tran Thanh[1, *], Tran Hoang Vu[1], Nguyen Duy Phuong[1], Do Son Tung[1], Cuong Nguyen-Van[1], Nguyen Van Cuong[2], Pham Ngoc Nam[1]

[1]School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam
[2]Faculty of Electronics and Telecommunications, Danang University of Science and Technology, Danang, Vietnam

Email address:
thanh.tran@hust.edu.vn (T. Thanh)

To cite this article:
Tran Thanh, Tran Hoang Vu, Nguyen Duy Phuong, Do Son Tung, Cuong Nguyen-Van, Nguyen Van Cuong, Pham Ngoc Nam. Enhance Performance in Implementing the Security of Partially Reconfigurable Embedded Systems. *American Journal of Embedded Systems and Applications.* Vol. 2, No. 1, 2014, pp. 1-5. doi: 10.11648/j.ajesa.20140201.11

Abstract: System Security means protecting confidentiality and integrity of input or output data to as well as the safety of the system. To meet stated security requirements, designers will need to add some special functions into the system. However, in several cases including high-throughput requirement, limited resource or trade-off between risks and costs, adding security functions should be taken into comprehensive consideration. This paper presents a method to enhance safety and update speed of the partially reconfigurable embedded systems based on FPGA is updated remotely via the Internet or from external storage devices such as a Compact Flash (CF) memory card.

Keywords: Reconfigurable, Bitstream Security, Embedded System, Partial Reconfiguration

1. Introduction

Field Programmable Gate Array (FPGA) technology provides the flexibility of on-site programming and re-programming of a system without going through re-fabrication with a modified design. Dynamic Partial Reconfiguration (DPR) takes this flexibility one step further, allowing the modification of an FPGA design by loading a Partial Bitstream (PB), usually a BIT file. After a Full Bitstream (FB) configures the FPGA, Reconfigurable Modules (RM) can be loaded to modify Reconfigurable Partitions (RP) in the FPGA without affecting the integrity of the applications running on those parts of the device that are not being reconfigured. In many cases it is useful to be able to swap out one or several of these subcomponents while the FPGA is still operating.

As shown in Fig. 1, functions are implemented in RP1, RP2 and RP3 can be modified by loading one of RM1 (A1.bit, A2.bit, A3.bit), RM2 (B1.bit, B2.bit) and RM3 (C1.bit, C2.bit, C3.bit, C4.bit) respectively. The logic in the FPGA design is divided into two different types: reconfigurable logic and static logic. The gray area of the FPGA represents the static logic; and the portion labeled RP represents reconfigurable logic. The static logic remains functioning and is completely unaffected by the loading of a RM into the RP. The functions of the reconfigurable logic can be replaced by the contents of the RMs (called also partial bitstreams).

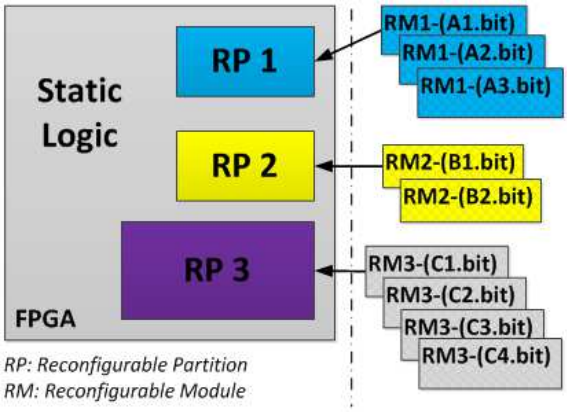

Figure 1. *The model of partial reconfigurable system.*

The partially reconfigurable systems based on offers many advantages [1]. These include:

- Reducing the size of a system that uses the FPGA to perform certain functions with the result that reduce

cost and power consumption
- Providing flexibility in the selection of algorithms or protocols available to an application
- Enabling new technologies in design security
- Improving the ability debugging on the FPGA
- Accelerating configurable computing

Besides the advantages of the flexibility of the partially reconfigurable system are the possible risks to the system as one of the modules may be malicious codes. In this paper, we present a method to protect the system by check the security and authenticity of the partial bitstream when they are remote updated from untrusted network.

In [2], the authors presented an implementation of bitstream encryption and authentication using AES-CBC encryption algorithm and SHA-256 authentication algorithm to compare with the authentication and encryption method AES-GCM. In [3], the authors separately implemented encryption algorithms including IDEA, DES, 3DES, Blowfish, AES, RC4, MD5, SHA-1 and SHA-256 on a Microblaze based embedded systems, with different architectures (MBlaze-A, MBlaze-B, MBlaze-C, MBlaze-D, MBlaze-E) depending on the combination of internal resources.

The main contribution of this paper is that we design a hardware AES-256 encryptor and a SHA-512 authenticator for optimal use of system resources while still meeting the performance requirements. Details of the AES (Advanced Encryption Standard) and SHA (Secure Hash Algorithm) algorithms are described in [4], [5].

The rest of the paper is organized as follows: Section 2 describes how to build a security system. Section 3 shows results of simulation and experiment. Finally, conclusions are given in Section 4.

2. Building System

2.1. General Diagram of the System

General diagram of the system is given in Fig. 2.

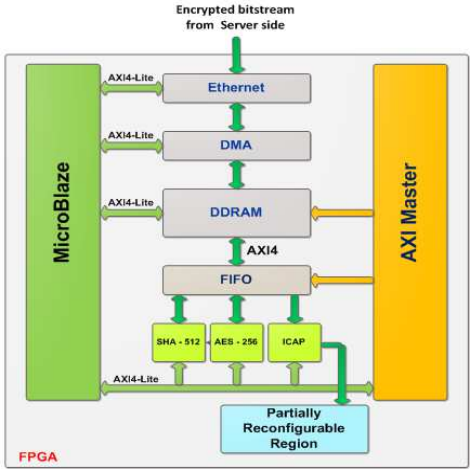

Figure 2. *General diagram of the system.*

2.1.1. The System Consists of the Blocks
- MicroBlaze soft-core processor controls blocks in the system and communicates with the Server through the Ethernet.
- AXI Master block controls the data transfer between DDRAM and the AES-256, SHA-512, ICAP blocks.
- Ethernet block receives and puts data from the server into the DDRAM.
- SHA-512 block executes authentication of the partial bitstream files.
- AES-256 block executes decryption of the encrypted partial bitstream files.
- ICAP is a control block for partial reconfiguration.

2.1.2. The System Operates as Follows
- With a new partial bitstream, Server will send a message to the User system.
- The MicroBlaze receives the message and sets the necessary registers in the Ethernet block to receive the bitstream.
- Thanks to the DMA block, data stream will come directly from the Ethernet block to the DDRAM without passing through the MicroBlaze. The MicroBlaze plays the role of the controller and sets the required parameters in the data-transfer process.
- After the data-transfer process, the MicroBlaze sets necessary parameters for SHA-512 and AES-256 to start operating. The input data of these blocks is loaded from the DDRAM through FIFO by the control of AXI Master.
- The MicroBlaze will compare output value of the SHA-512 authenticator to the message authentication code that received from the server. If the results are the same, the MicroBlaze will write the chipSelect signal to allow ICAP operating, and the process of reconfiguration will begin.

2.2. Building Functional Hardware Blocks

The function blocks are built using Xilinx XPS (Xilinx Platform Studio) version 14.1 software, and the hardware flatform is Virtex-6 FPGA.

2.2.1. AXI Ethernet Core

The main component of the data-transfer block is a LogiCORE™ IP AXI Ethernet core [6], Fig. 3. This core is supported by Xilinx. It implements a tri-mode (10/100/1000 Mb/s) Ethernet MAC, and provides a control interface to internal registers using a 32-bit AXI4-Lite interface subset. This AXI4-Lite slave interface supports single beat read and write data transfers. The transmit and receive data interface is through the AXI4-Stream interface.

This core is based on the Xilinx hard silicon Ethernet MAC in Virtex-6 devices and provides a soft Ethernet MAC option for supported devices.

2.2.2. AXI DMA Core

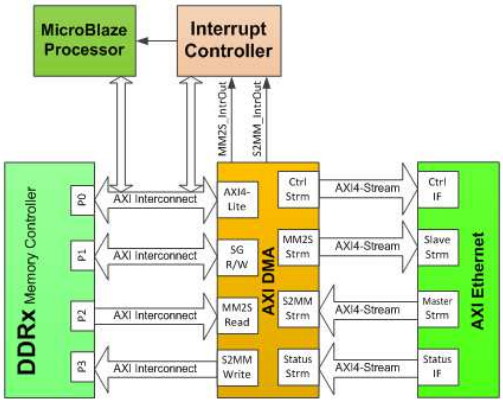

Figure 3. *Typical MicroBlaze processor system.*

AXI DMA (Advanced eXtensible Interface Direct Memory Access) core is a soft Xilinx Intellectual Property (IP) core for use with the Xilinx® Embedded Development Kit (EDK) and the CORE Generator™ tools [7]. The AXI DMA engine provides high-bandwidth direct memory access between memory and AXI4-Stream-typetarget peripherals. Its optional scatter gather capabilities also off-load data movement tasks from the Central Processing Unit (CPU). Initialization, status, and management registers are accessed through an AXI4-Lite slave interface, which is suitable for the Xilinx MicroBlaze™ microprocessor.

The AXI DMA IP core provides high-bandwidth direct memory access between the AXI4 memory mapped and AXI4-Stream IP interfaces. AXI Master MM2S used to read data from memory and AXI Slave S2MM used to write data into memory from the blocks as shown in Fig. 3.

2.2.3. AXI Master Core

AXI Master [8] core is added to system by using Creat and Import Peripheral Wizard tool in the Xilinx XPS version 14.1. This core provides a bidirectional interface between a User IP core and the AXI4 interface standard. It also provides a data-transfer interface between peripheral blocks and DDRAM through a FIFO.

However, it is difficult to control the data read and write process of both hardware and software in single FIFO. Our solution made using two FIFOs includes the first FIFO controlled to write by software and to read by hardware, and the second FIFO controlled in the opposite direction, as shown in Fig. 4.

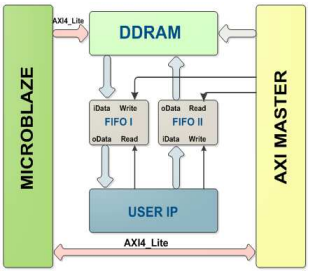

Figure 4. *The system with two FIFOs.*

2.2.4. SHA-512 Core

The SHA-512 core takes care the authentication of the partial bistream file. It was built by FSMD model [9] with two blocks, as shown in Fig. 5. Controller block is SHA_512_Controller, and Datapath block is SHA_512_Function.

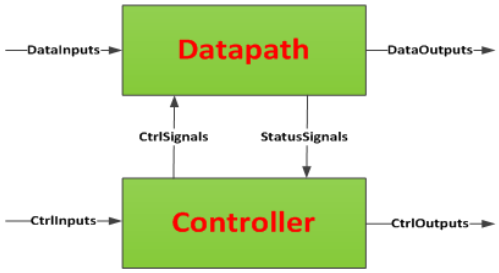

Figure 5. *FSMD model.*

SHA_512_Controller handles received status signals from the SHA_512_Function and controls the communication process with DDRAM through FIFOs.

SHA_512_Function performs the functions of the SHA-512 algorithm. It consists of three small blocks: a 4-bit counter, an 8-bit counter and a ROM memory to store constants and intermediate functions.

2.2.5. AES-256 core

The AES-256 core is also built in the FSMD model, which consists of two major blocks. AES_256_Controller block plays the role of state FSM controller and AES_256_Function block plays the role of data block (datapath).

The AES_256_Controller receives status signals as input signals then decides to give the control signals for the AES_256_Function. It also controls the data exchange from the FIFO through read and write signals.

The AES_256_Function consists of four small blocks taking four different functions. Such blocks include Inverse SubBytes, Inverse ShiftRows, AddRoundKey and Inverse MixColumn. It also generates a KeyExpand block to perform the AddRoundKey in the cycle.

We designed cores of the SHA-512 authenticator and AES-256 decryptor by making them into the User IP core from VHDL source code and integrating them into the system using the Xilinx XPS tool.

3. Prototype System and Analysis of Results

3.1. The Prototype System

The prototype system is shown in Fig. 6. It consists of a PC and a Xilinx Virtex-6 FPGA ML605 evaluation board connected via Ethernet and UART interface. The PC plays the role of a Server, which is in charge of managing data as partial bitstream files and User profiles. The Xilinx Virtex-6 FPGA ML605 evaluation board acts as a device with modules which can be partially reconfigured.

Figure 6. *The prototype system.*

3.2. The Simulation Results

Before implementing of security algorithms to hardware, we used the simulation tool of Xilinx Integrated Software Environment (ISE) to check accuracy of the proposed methodology. Results are described as follows.

Received Message Authentation Code (MAC) of the partial bitstream file, Led.bit, through the SHA-512 authenticator is "4f681e0bd53cda4b5a2041cc8a06f2eab de44fb16c951fbd5b87702f07aeab611565b19c47fde305871 77ebb852e3971bbd8d3fd30da18d71037dfbd98420429". It is equal to the MAC done on the server, as shown in Fig. 7.

Fig. 8 shows the simulation result from the implementation of the AES-256 decryptor. We used a 256-bit key, K = 08090A0B0D0E0F1012131415171819 1A1C1D1E1F21222324262728292B2C2D2E, and encryted data is C = "080 E9517EB1677719ACF7280860 40AE3...". Received plaintext is M = "069A007FC 76A459F98BAF917FEDF9521...". This result is similar to the original data into an encryptor on the server.

Figure 7. *The simulation results of the SHA-512.*

Figure 8. *The simulation result of the AES-256.*

3.3. The Experimental Results

Fig. 9 shows a visualization of the results of partial reconfiguration for FPGA by the simple LED SWITCH on Xilinx Virtex-6 FPGA ML605 board with encrypted file, Led.bit, stored on the server.

Figure 9. *Partial Reconfiguration of the LED SWITCH Module.*

3.4. Evaluating of Results

The experimental and simulation results showed that adding security and authentication function to the system design is absolutely realizable. Based on the results of the Table 1, we can see that the hardware resources be used for the AES-256 and SHA-512 only account for 5% of the system resources. This is a good result to consider security implementation in the partially reconfigurable embedded systems.

By virtue of using the AXI Master with two FIFOs, the performance of the AES-256 decryptor and SHA-512 authenticator improves approximately four times, as shown in Table 2. This result is significant for the continuous increase in the FPGA size.

Table 1. *Hardware utilization of SHA-512 and AES-256.*

Resources	IP Core	Used	Available	(%)
Register	SHA-512	2246	301440	~1%
	AES-256	3096	301440	~1%
LUTs	SHA-512	2299	15720	~1%
	AES-256	3751	15720	~1%
Slices	SHA-512	843	37680	~2%
	AES-256	1293	37680	~3%

Table 2. *Performance of SHA-512 and AES-256.*

IP Core	Throughput(Mbps) Without AXI Master	Within AXI Master
SHA-512	738 Mbps	3099 Mbps
AES-256	1264 Mbps	4929 Mbps

4. Conclusions

In this paper, with the proposed method, all the partial bitstream files to ensure always be encrypted and authenticated when loading into the system. Adding the AES-256 decryptor and the SHA-512 authenticator makes the system more secure, and does not increase system resources significantly.

The proposed model for DPR systems offers improved security and convenient reuse of external third-party IP cores. It also protects the IP Digital Rights Management as well as system security and integrity in embedded design environment with multi-party participants.

References

[1] Xilinx Inc., "Partial Reconfiguration User Guide," User guide UG702 (v14.5), 2013. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/ug702.pdf.

[2] Y. Hori, A. Satoh, H. Sakane, and K. Toda, "Bitstream Protection in Dynamic Partial Reconfiguration Systems Using Authenticated Encryption," IEICE transactions on Information and Systems, Vol.E96-D, No. 11, pp. 2333–2343, November 2013.

[3] I.Gonzalez and F.J. Gomez-Arribas, "Ciphering algorithms in MicroBlaze-based embedded systems," in IEEE Proceedings - Computers and Digital Techniques, 2006, pp. 87–92.

[4] NIST., "Announcing the Advanced Encryption Standard (AES)," National Institute of Standards and Technology, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[5] NIST., "Secure Hash Standard (SHS)," National Institute of Standards and Technology, 2012. [Online]. Available: csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.

[6] Xilinx Inc., "LogiCORE IP AXI Ethernet (v3.01a)," Product Specification, 2012. [Online]. Available:xilinx.com/support/documentation/ip_documentation/axi_ethernet/v3_01_a/ds759_axi_ethernet.pdf.

[7] Xilinx Inc., "LogiCORE IP AXI DMA (v5.00.a)," Product Specification, 2011. [Online]. Available:xilinx.com/support/documentation/ip_documentation/axi_dma/v5_00_a/pg021_axi_dma.pdf.

[8] Xilinx Inc., "Xilinx DS844 LogiCORE IP AXI Master Burst (v1.00.a)," Product Specification, 2011. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/axi_master_burst/v1.00a/ds844_axi_master_burst.pdf.

[9] N. Kavvadias, V. Giannakopoulou, and K. Masselos, "FSMD-Based Hardware Accelerators for FPGAs," In Embedded System - Theory and Design Methodology. InTech, 2012, pp. 143–166.