Integration of AES on Heterogeneous Many-Core system

Hassan Anwar, Masoud Daneshtalab, Masoumeh Ebrahimi, Marco Ramirez, Juha Plosila, Hannu Tenhunen
Department of Information Technology, University of Turku, Turku, finalnd
{hasanw, masdan, masebr, maanrl, juplos, hanten}@utu.fi

Abstract — **Increasing in the transistor density in a single chip makes it possible for many-core systems to utilize design space for implementing complex embedded systems. In this paper, we propose an architecture for heterogeneous many-core system to integrate block cipher algorithm which is based on Advanced Encryption Standard (AES). In order to implement AES as a crypto-core along with heterogeneous many-core system two different approaches are proposed. In the first approach, the platform is composed of an AES module, a crypto-core, a network interface, and an internal memory which are managed through a controller. In the second approach, the platform is composed of a Direct Memory Access (DMA), network interface, an internal memory, and a microprocessor in which the AES module is integrated as a crypto-core. Both approaches have been analyzed and compared in terms of area overhead and performance.**

Keywords—Many-Core; Field Programmable Gate Array; AES; Network Interface; Crypto-Core.

I. INTRODUCTION

Multiprocessor systems-on-chip (MPSoC) have been proposed [1]-[2]. In order to process applications at faster rates which are not possible by a single core processor. In order to provide data security for applications running on MPSoC we incorporate MPSoC with Advanced Encryption Standard (AES). The AES is works as a crypto-core in order to provide the data security to applications which are supposed to be run on heterogeneous many-core systems. Field Programmable Gate Arrays (FPGAs) is a suitable choice for implementing cryptographic algorithms on hardware. For instance, servers that need to handle lots of encrypted authentications are benefited by using FPGAs. The AES is a very widespread symmetric-cryptography algorithm for encrypting the data. FPGAs offer the required performance for this algorithm. Besides the performance and speed, there are other characteristic of FPGAs (e. g. device utilization) that make them suitable choices for cryptography [3]. The main purpose of the cryptographic algorithm is to provide security. The Data Encryption Standard (DES) [4] has been used as a cryptographic algorithm for last several years but it is replaced by the Rijndael algorithm due to its shorter key length. The Rijndael algorithm has become as a standard in cryptography which is called Advanced Encryption Standard [5]. Encryption is a transformation technique to change the user data (known as plain text) to an unreadable form called cipher text. The hardware implementation of the crypto algorithm which is associated with keys is by nature more secure in terms of unauthorized usage [6]. FPGAs provide a suitable platform for the hardware implementation of cryptographic algorithms because of their re-programmable capability and their ability to modify design at any time. FPGAs offer the best solution for researchers to implement and test the designs.

When the Rijndael algorithm has become as a standard encryption algorithm, many hardware implementations based on FPGA and Application Specific Integrated circuit (ASIC) have been proposed [7-12]. ASIC provides low power design but the design time and time to market are very high and it has the lack of flexibility.

Number of transistors on a processor core is intensely increasing but clock speed does not grown at the same speed [13]. The increased number of transistor on a processor core makes it possible to design a complex and heavily computational systems on a single chip called System-on-Chip (SoC). The SoC contains a processing element (PE), memory block IP core, and communication architecture. SoC integrated with several processors in a same chip is named MPSoC. In this paper, we present two different approaches for FPGA based MPSoC platforms name AXI-based MPSoC (AXIM) [14], with AES module on a tile. The tile is composed of Network Interface (NI), Controller, AES, and internal memory for first approach (see Fig. 3), and in the second approach tile is composed of NI, plasma processor [15], AES, Direct Memory Access (DMA), and internal memory (see Fig. 4). The plasma processor used in the second approach is a three stage pipelined 32-bit Reduced Instruction Set computer (RISC) microprocessor. In order to offer encryption and decryption, the AES module works as a crypto-core with the processor module. This paper is organized as follows. Section II presents the AES algorithm. Section III describes the FPGA implementation of AES. The AXIM platform is described in Section IV. The implementation approach of the tile (proposed design) is presented in Section V. Experimental results are given in Section VI while Section VII concludes the paper.

II. AES ALGORITHM

As it is already mentioned, an AES has been selected as an encryption standard in November 2001 which is based on the

Rijndael algorithm [1]. AES utilizes symmetric cipher blocks, supporting different data lengths of 128, 192 and 256 bits. It also has the ability to support different key sizes of 128, 192 and 256 bits. Encryption and decryption depends on the number of rounds, i.e. 10 (requires 128 bits), 12 (requires 192 bits), and 14 (requires 256 bits). The AES algorithm comprises of four different steps as: key addition, S-box substitution, Shift Rows, and MixColumn (see Fig. 1).

The S-box substitution is the only non-linear transformation which comprises of two steps. The first step is the multiplicative inverse of input bytes in which an affine transformation can be applied on it. There are two different ways to implement S-box entries as using look up tables or computing mathematically. The purpose of Shift Rows and MixColumn are to create diffusion that is a linear operation.

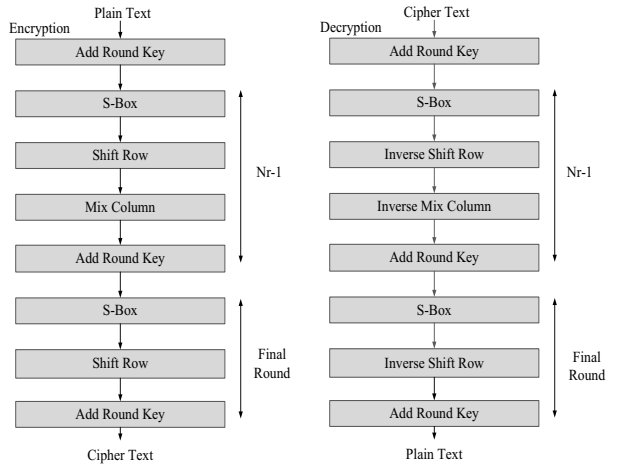

Fig. 1. AES Algorithm.

III. FPGA IMPLEMENTATION OF AES

The efficient implementation of the AES algorithm on FPGA is being under discussion from last several years in terms of throughput, resource utilization, and high speed. The main reason to choose FPGA for the implementation of the cryptographic algorithms is because of FPGA permits to change the design with almost no additional time. The cost and a design cycle is also very low for FPGA based design. An FPGA-based AES implementation is presented in [6] while several other high speed Implementations have been explored in [7-12], in which the speed ranged from Mbps to several Gbps. The pipelining in the Advanced Encryption Standard is used to get a higher throughput and a higher speed as multiple rounds of AES can be handled concurrently. The pipelined AES have been proposed to achieve high throughput which can achieve the throughput of around 20.3 Gbps [16-18].

IV. AXIM PLATFORM

The AXIM platform [19] has three main components: a cluster of processing elements (CPE), the Advance eXtensible Interface (AXI) subsystem [20] and an embedded controller.

As shown in Fig. 2, the AXIM platform uses the AXI interface in order to provide linkage between AXI and CPE. The AXI subsystem is constituted by a set of AXI controllers, external DDR memory, Ethernet, FLASH memory, and USB host. A keystone of the platform is the NoC-AXI interface. It creates the communication channel between the CPE and AXI bus. The internal memory receives data from the external memory through NoC-AXI interface. NoC-AXI interface are composed of following components:

Network interface [21]: lies between a tile and NoC to handle the data flow control between router and the rest of the internal units.

AXI units: AXI bus handles read and write transactions and both are independent allowing simultaneous execution.

Internal memory unit: stores data need to be processed by the micro-processor.

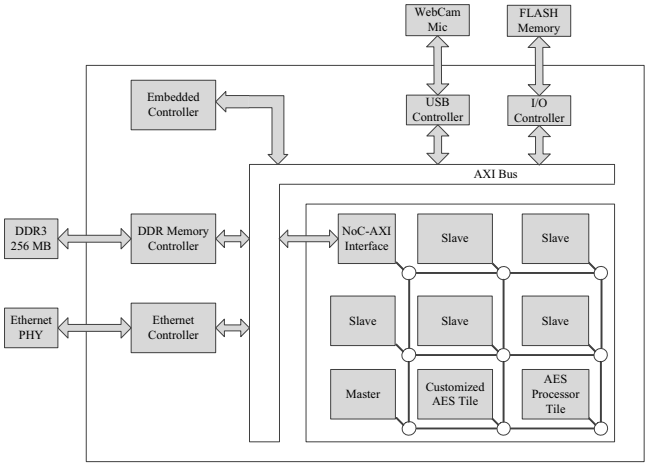

Fig. 2. AXIM platform.

In this work, we equip AXIM with the customized AES-based crypto module. This enables a heterogeneous many-core platform to be capable of accelerating the AES-based crypto requests [22]. There are two different design approaches. For the first approach, we develop a customized AES-based crypto tile, shown in Fig. 3. The main component of this tile is the controller which is designed such that to communicate with the network interface [23]. The controller is designed in such a way to read the data from the memory for AES operations and also used to write the AES output back to memory. The output of the AES module is called cipher data. This cipher data needs to be write back into the memory which is also accomplished by the controller. The second approach is to integrate the AES module with the plasma processor in a tile, which is depicted in Fig. 4. In this approach, we use the plasma processor and DMA. The processor is used to generate the crypto-instruction for the AES module in order to perform the AES operations. The DMA module is used to access contents of the internal memory to load/store the plain or cipher text. The on-chip network exploits static XY routing [24].

V. IMPLEMENTATION

Fig. 3 shows the basic design for the first approach; the external memory contains the data which is supposed to be

encrypted/decrypted by the AES. The controller is used to control all the operations among the network interface, internal memory, and the AES module. Initially, the controller gets the data for the internal memory through the network interface from the external memory. When the data is stored in the internal memory, the AES module will be activated. Once the process is finished by the AES module and output gets ready, the cipher data will write back to the internal memory.

For the second approach we integrate the AES module with the plasma processor, presented in Fig. 4. In this case, the internal memory accesses the data through DMA and the AES module is activated by the plasma processor through dedicated instructions called crypto-instruction.

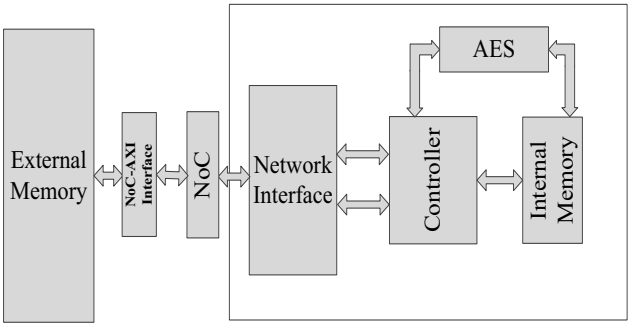

Fig. 3. Customized AES-based Crypto Tile.

When the AES module is activated by the plasma processor, it starts receiving the data from the internal memory and performs encryption or decryption on it. Then the output of the AES module will write back to the internal memory.

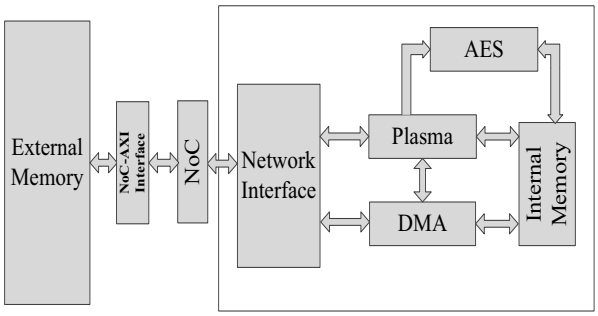

Fig. 4. AES-based Crypto Co-Processor Tile.

In order to write back the AES output, called cipher data, to the internal memory, first this data is stored in four 32-bit registers. Then the registers' data (cipher data) will write back to the internal memory. The controller logic used in customized AES-based crypto tile is based on the state machine logic, which is used for the communication between the internal memory and the AES module. Furthermore, the state machine is used to copy data (cipher data) from the register to the memory. Similarly, the state machine is used to copy data (plain data) from the internal memory to the register. The AES module requires 128 bits of data so that the state machine receives 32 bits of data on each clock cycle and stores them on the registers.

VI. Experimental Results

Integration of the AES module as a crypto-core within the AXIM platform is one of the main idea of this work. Two different design approaches are considered. The platform was built using VHDL and its main features were verified through simulation with Modelsim. The design approaches are synthesized for Xilinx Virtex6 ML605. Xilinx ISE 14.4 is used for synthesizing the designs and generating statistics about resource utilization shown in Table I and Table II.

We compute the latency for both approaches as well. For experimental set up, we store 100 requests in an external memory and measure the latency. The latency is the time takes for the system to complete the encrypt/decrypt requests stored in the external memory. The latency observed for the customized AES-based crypto tile is about 815 ns. The latency for the system based on the second approach is about 1235 ns. The operating frequency of the system is 100 MHz. The first approach is simpler than the second one as the plasma processor and DMA are replaced by the controller for the first approach. The hardware resource utilization for the first approach is lower than the second approach because the DMA module and the plasma processor imposes extra overhead for each tile in the system. Also, the number of clock cycles required by DMA is more than the number of clock cycles required by the controller to access the memory.

The DMA module used in the second approach is slower than the controller used in the first approach because DMA will take extra clock cycles to decide whether the data required by the memory or by the processor which increases the overall latency. However, in the first approach the controller access data in one clock cycle. In brief, the first design approach not only reduces the hardware resources but also increases the performance significantly. We also calculate latency for the different number of requests for both approaches shown in Fig. 5. Latencies for both approaches increase as the number of requests grows. Latency for the second approach is greater than the first approach because of the extra overhead generated by DMA and the plasma processor.

VII. Conclusion

This work presented a method to incorporate the AES module as a crypto-core within AXI-based MPSoC platform. We proposed two different approaches for the integration of the crypto-core with heterogeneous many-core system. The current work is focused on the design and implementation of crypto-core inside with a many-core system. The main purpose of this work is not only to provide the platform to utilize the many-core resources but also provide the security to many-core systems.

Table I: Resource utilization for customized AES-based Crypto Tile.

Functional Unit	**LUTs**
Network Interface	237
Controller	75
AES	3112
NoC-to-AXI Interface	1,503
Internal Memory	457
Total	5384

Table II: Resource utilization for AES-based Crypto Co-Processor Tile.

Functional Unit	LUTs
Network Interface	237
Plasma Processor	2935
Direct Memory Access	195
AES	3112
NoC-to-AXI Interface	1,503
Internal Memory	457
Total	8439

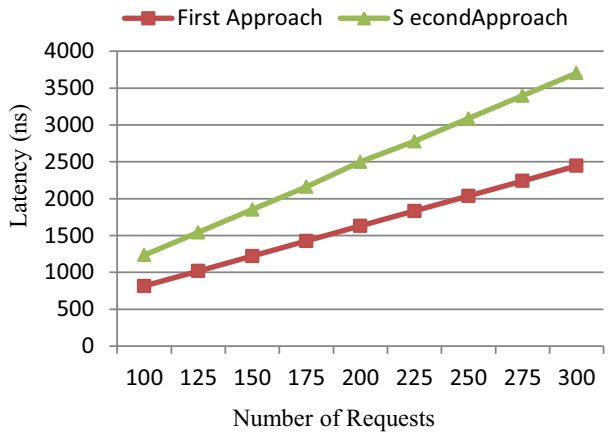

Fig. 5. Latencies for different number of requests for both approaches.

REFERENCES

[1] A. Shabbir, A. Kumar, B. Mesman and H. Corporaal, "Distributed resource management for concurrent execution of multimedia applications on MPSoC platforms," ICSAMOS, pp. 132-139, July 2011, Greece.

[2] M. Fattah, M. Daneshtalab, P. Liljeberg and J. Plosila, "Exploration of MPSoC Mointoring and Management systems," in proceedings of IEEE International Symposium on Reconfiguarble communication-centric Systems-on-chip (ReCoSoC), pp. 1-3, June. 2011, France.

[3] 40Gbit AES Encryption using OpenCL and FPGAs, http://nallatech.com/images/stories/technical_library/whitepapers/40_gbit_aes_encryption_using_opencl_and _fpgas_final.pdf.

[4] B. Schneier, Applied Cryptography, Wiley, New York, 1996.

[5] National Institute of Standard and Technology (USA, Advanced Encryption Standard). FIPS 197, Available at, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, September 1999.

[6] S. M. Yoo, D. Kotturi, D. W. Pan and J. Blizzar, "An AES crypto chip using a high-speed parallel pipelined architecture," Elsevier, Microprocessor and Microsystem, vol. 29, pp. 317-236, Jan. 2005.

[7] P. Chodowiec, "Comparison of the hardware performance of the AES candidates using rconfigurable hardware," Master's thesis, George Mason University, March 2002.

[8] H. Kuo and I. Verbauwhede, "Architectural optimization for 1.82 Gbits/sec VSLI implementation of the AES Rijindael algorithm," Cryptographic Hardware and embedded systems (CHES'01), vol. 2162, pp. 51-64, Springer-Verlag, 2001.

[9] N. Sklavos and O. Koufopavlou, "Architecture and VLSI implementation of the AES-proposal Rijndael," IEEE Trans. Computers, vol. 51, pp. 1454-1459, 2002.

[10] P. R Schaumont, H. Kuo and I. M. Verbauwhede, "Unlocking the design secrets of a 2.29 Gb/s Rijndael processor," ACM Conference on Design Automation (DAC 2002), pp. 634-639, June. 2002, USA.

[11] S. Morioko and A. Satoh, "A 10 Gbps full-AES crypto design with twisted BSS s-box architecture," 21st IEEE International Conference on Computer Design VLSI in Computers and Processors, pp. 98-103, Sept. 2002, Germany.

[12] U. Mayer, C. Oelsner and T. Kohler, "Evaluation of different Rijndael implementaion for high end servers," IEEE International Symposium on Circuits and Systems, vol. 2, pp. 348-351, May. 2002, USA.

[13] E. W. Wachter, A. Biazi and F. G. Moraes, "HeMPS-S: A homogenous NoC-based MPSoCs framework prototyped in FPGAs," 6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip(ReCoSoC), June. 2011, France.

[14] ARM, "AMBA AXI and ACE Protocol specification," Available on-line.

[15] N. Kranitis, G. Xenoulis, D. Gizopoulos, A. Paschalis and Y. Zorian, "Low-Cost Software-Based Self-Testing of RISC Processor Cores," Proceedings of the conference on Design, Automation and Test in Europe, pp. 10714, vol. 1, Mar. 2003, Germany.

[16] N. C Iyer, P. V Anandmohan , D. V Poornaiah and V. D Kulkarni, "High through put, low cost, fully pipelined architecture for AES Crypto Chip," Annual IEEE India Conference, pp. 1-6, Sept. 2006, India.

[17] Y. Zhang and X. Wang, "Pipelined implementation of AES encryption based on FPGA," IEEE International Conference on Information theory and Information Security, pp. 170-173, Dec. 2010, Beijing.

[18] I. Verbauwhede, P. Schaumont and H. kuo, "Design and performance testing of a 2.29-Gb /s rijndael processor," IEEE J. Solid State Circuits, vol. 38, pp. 569-572, 2003.

[19] M. Ramirez, M. Masoud, J. Plosila and P. Liljeberg, "NoC-AXI Interface for FPGA-based MPSoC Platforms," 22nd International Conference on Filed Programmable Logic and Applications (FPL), pp. 479-480, Aug. 2012, Norway.

[20] ARM. AMBA AXI and ACE Protocol Specification. Available on-line.

[21] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, H. Tenhunen, "A High-Performance Network Interface Architecture for NoCs Using Reorder Buffer Sharing," in Proceedings of 18th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP), pp. 547-550, Feb. 2010, Italy.

[22] J. Carabaño, F. Dios, M. Daneshtalab, and M. Ebrahimi, "An Exploration of Heterogeneous Systems," in Proceedings of 8th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), pp. 1-7, July 2013, Germany.

[23] M. Ebrahimi, M. Daneshtalab, N. Sreejesh, P. Liljeberg, Juha Plosila, H. Tenhunen, "Efficient Network Interface Architecture for Network-on-Chips," in Proc. of 27th IEEE Norchip, pp. 1-4, Nov. 2009, Norway.

[24] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "LEAR – A Low-weight and Highly Adaptive Routing Method for Distributing Congestions in On-Chip Networks," in Proceedings of 20th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP), pp. 520-524, Feb. 2012,Germany.