Dynamically Reconfigurable Resource Efficient AES Implementation for IoT applications

Abdelrahman M. Ruby[1], Shady M. Soliman[1], and Hassan Mostafa[2,3]

[1]B.sc of Electronics Engineering, German University in Cairo, Cairo, Egypt.
[2]Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt
[3]Nanotechnology Department at Zewail City for Science and Technology, 6th of October, Giza 12578, Egypt
abdelrahmanroubi@hotmail.com, shady_soliman2002@yahoo.com, hmostafa@uwaterloo.ca

***Abstract*—Internet of Things (IoT) is the ability of things to share useful data among each other. It is becoming one of the most crucial technologies of our generation, however, one of its biggest challenges is security. In this paper, a design is proposed using Advanced Encryption Standard (AES) and the Dynamic Partial Reconfiguration (DPR) feature of the FPGA to tackle the security problem. AES-128 is used with 128-bit input data and 128-bit key. DPR is a new feature that allows utilizing the same hardware for different functions, which minimizes area and power needed by a system. The variants of the DPR are one round of encryption and one round of decryption. The proposed design offers low hardware and low power consuming cryptographic algorithm. The average reduction in resources consumed is 33% for encryption and 29% for decryption and energy utilization is decreased by 43.75%. The proposed work is tested on ZC702 evaluation board, synthesized and implemented using Vivado 2015.2.**

***Index Terms*—AES, DPR, Encryption, Decryption, IoT, FPGA**

I. Introduction

IoT according to IEEE is "A network of items—each embedded with sensors—which are connected to the Internet" [1]. It is rapidly imposing itself in today's world empowering each device with the needed sensors, microcontrollers and communication capabilities to share useful data among each other. This novel paradigm is expected to ease our lives with applications such as domotics, e-health, enhanced learning, automation, intelligent transportation, and many other industrial systems [2] [3].

IoT technologies and applications are still in their infancy. This implies the existence of many challenges for its commercial use, such as: technology, standardization, area, power, and security [3]. Since the size of an IoT device is usually very small, limited number of resources can be equipped, confining the hardware resources. The power constraint of IoT devices stems from the fact that a large percentage of IoT devices are powered from either small batteries, or renewable energy sources. Considering that a device with low power is more significant than a secure one, sometimes security and ciphering are sacrificed for power saving modes. Resulting in a security challenge epitomized in the demanded low power cipher [4].

These challenges could be parleyed by using a FPGA. Field Programmable Gate Array (FPGA) are widely used nowadays in many applications that previously needed Application Specific Integrated Circuit (ASIC) prototyping [5]. Besides the simplicity of implementation, it provides a set of tools that optimizes designing an application on it. Dynamic reconfiguration supported by modern FPGA families, is among those tools. Most of the IoT security challenges can be overcame by using dynamic reconfiguration, as it permit the use of the same hardware resources for different functions, which in its turn reduce the area and power resources consumed.

The objective of this paper is to deliver a highly secure and less resource (area and power) consuming encryption/decryption algorithm to be deployed on IoT devices. To achieve this, a technique for implementing Rijndael cipher algorithm (also known as AES) through Dynamic Partial Reconfiguration (DPR) of an FPGA is developed. Non-pipelined, looped (unrolled) architecture AES-128 (128-bit key length) with small area is implemented. While DPR variants are one round of encryption and one round of decryption.

Relatable techniques and approaches were discussed in other papers. In [5], Burman et al. deployed DPR on AES, while the key size (128-bit, 192-bit, or 256-bit) is the reconfiguration parameter. Khatib et al. [4] proposed a power adaptive encryption design that chooses -using DPR- a suitable cipher depending on the available power.

This paper is organized as follows: Section II describes the AES algorithm, the DPR, and the Linear Feedback Shift Register (LFSR). Section III presents the proposed work and elucidate the implementation. Section IV presents the results and discussion followed by the conclusion in section V.

II. Background

This section explains the AES algorithm, the DPR of an FPGA, and the LFSR.

A. Advanced Encryption Standard

Rijndael algorithm is a symmetric encryption algorithm, chosen by the National institute of Standards and Technology (NIST) on 2001 to be used as the Advanced Encryption Standard [6]. Symmetric means that encryption and decryption uses the same cipher key. It takes a 128-bit plain/cipher text as its input, that is sorted in a 4x4 matrix of bytes (called the state), and output a 128-bit cipher/plain text. The cipher key used can vary in size between 128-bit, 192-bit, and 256-bit (longer key length indicates higher security), resembling

978-1-7281-3320-1/20/$31.00 ©2020 IEEE

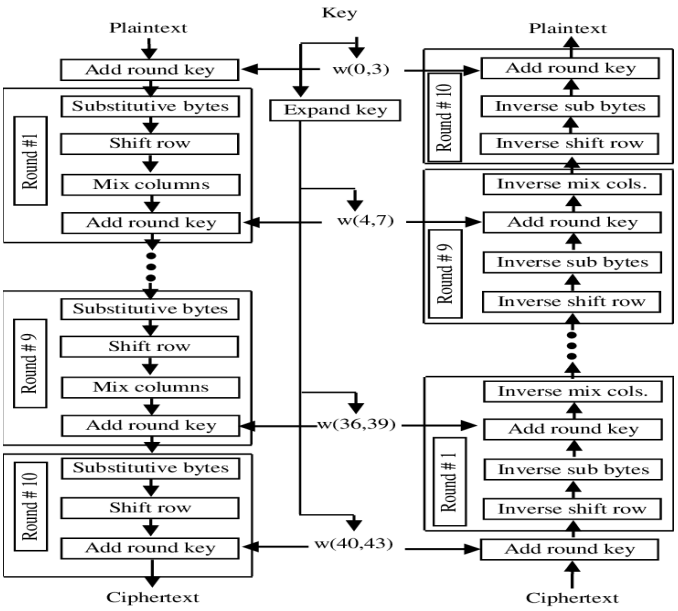

Fig. 1. The transformation flow of encryption and decryption of the AES algorithm.

different versions of AES, namely AES-128, AES-192, and AES-256. The state undergo N rounds depending on the key length used. N is 10, 12, and 14 rounds for key sizes of 128-bit, 192-bit, and 256-bit respectively. These rounds have functions (transformations) that causes the diffusion, each with a 4x4 matrix (of bytes) as its input and output. Figure 1 elaborates the flow of the AES algorithm, first the state goes through the initial round that has a single transformation (AddRoundKey). The state then undergoes N-1 identical rounds that has the same 4 transformations ((inv)SubBytes, (inv)ShiftRows, (inv)MixColumns, and AddRoundKey). The final round is the same but for the (inv)MixColumns.

B. Dynamic Partial Reconfiguration

Reconfiguration is the ability of an FPGA to be reprogrammed through changing the resources (CLBs, interconnects, and IO pins) functionality. Reconfiguration can either be static; where the device has to be reset to be reprogrammed, or dynamic; that allows the device to be reconfigured at runtime. Dynamic reconfiguration can be classified into total; where the whole design is replaced with a new one. And partial; that enables changing the functionality of a preselected area (module) of an FPGA during runtime, without interrupting the rest of the FPGA's operation. The design is partitioned according to the DPR design flow to a static part and a dynamic part. The dynamic part accommodate a set of Reconfigurable Partitions (RPs). Each RP has a set of substitute modules called Reconfigurable Modules (RMs) that can be swapped during runtime, while the static part remains unchanged.

PR controllers are used to control the DPR by providing an interface for transferring partial BIT files from an external or internal memory to the FPGA internal configuration port (ICAP or PCAP). That in its turn transfer data

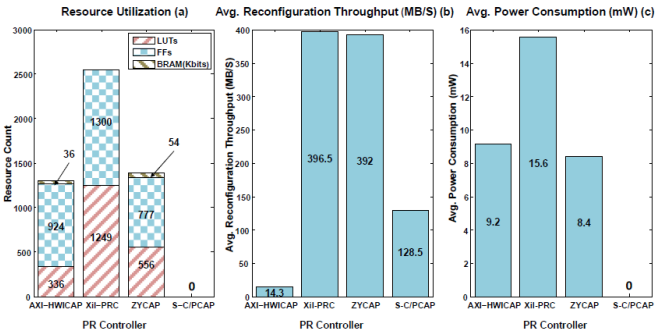

Fig. 2. Comparison between different PR controllers [8].

to the FPGA configuration memory for reconfiguration [7]. It aims to decrease the reconfiguration time to enhance the reconfiguration throughput. In [8], four PR controllers namely HWICAP, Xilinx Partial Reconfiguration Controller (PRC), ZYCAP, and PCAP are compared with respect to resource utilization, reconfiguration throughput, and power consumption. The comparison results are shown in Figure 2. Since in IoT, the consumed hardware and power resources are the most important aspects, AXI-HWICAP is used in this paper for better resource utilization. It is an ICAP controller that enables the configuration memory to be accessed by an embedded microprocessor.

C. Linear Feedback Shift Register

A LFSR is a simple shift register acting as a Pseudo Random Number Generator (PRNG), with an initial value called the seed. The value to be shifted in is a linear function of the current value. Usually the linear function is an exclusive-or (XOR) of some specific bits of the current value as shown in figure 3.

Now, that we have briefly explained each of the techniques used in this paper, the next section discusses the proposed work.

III. Proposed Work

The proposed work intends to offer a security solution for IoT applications. This is done through providing low power and low area consuming, highly secure cryptography algorithm. IoT devices are usually small and operated by limited energy sources, so area and power are their main challenges. On the contrary, the data exchanged among IoT devices are

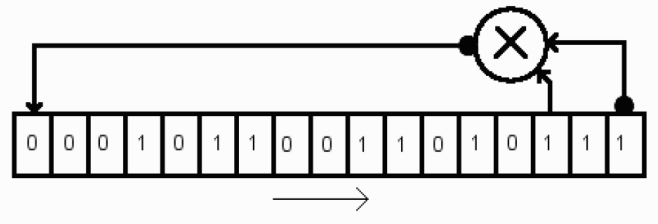

Fig. 3. Operation of the LFSR.

usually small of size, hence, high throughput is not much needed. In order to have a significant decrease in resource consumption, in this paper, Dynamic Partial Reconfiguration (DPR) is utilized with the AES algorithm using the AXI-HWICAP PR controller. The next subsections discuss the implementation of the project.

A. AES with LFSR

The top module contains three main modules that are responsible for ciphering/deciphering. The first is the Key Scheduling module, in charge of generating the 10 sub-keys used in the AddRoundKey transformation. The Key Scheduling module does the same function for encryption and decryption. The second is the Encryption module, it performs one round of the encryption process of the AES algorithm. The Decryption module that performs one round of the decryption process of the AES algorithm comes in as the last module. A diagram for the system is shown in figure 4.

The top module first triggers the Key Scheduling module and saves the 10 generated sub-keys to a DRAM, while the other two modules are halted to save energy. Once it finished generating the keys, the module is stopped to save energy. On the fly calculation of sub-keys were not used as power is targeted, not throughput. The top module then selects whether to encrypt or decrypt, depending on an input from the user.

Initially, the first round of encryption or decryption is done separately (AddRoundKey only). The top module then nearly performs the same sequence for the encryption and decryption module. It manipulates the input to the encryption or decryption module by re-feeding its output to it with a new sub-key, to yield the right round's output. When it's the last round, a signal is triggered that skips the (inv)MixColumns transformation. After displaying the 128-bit output, if a new input is present, and the key is changed, the Key Scheduling operation is reset. If the key is not changed, the algorithm generates sub-keys for a key generated by the LFSR with the initial key as its seed.

A LFSR is used to enhance the security of the AES algorithm, by generating pseudo random versions of the input key. First, it XOR the input key with a NONCE defined in the LFSR module, then it performs the LFSR function. As long as the key is the same, this operation keeps repeating by feeding the previously generated random key to the LFSR.

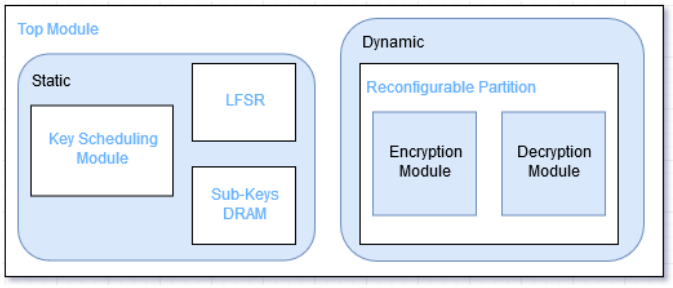

Fig. 4. System diagram showing the static and dynamic parts of reconfiguration.

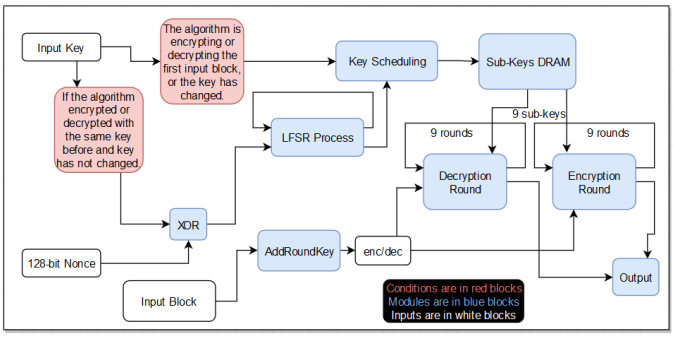

Fig. 5. Block diagram explaining the AES with LFSR module.

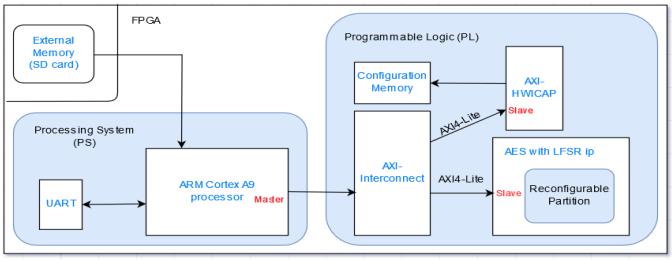

Fig. 6. Detailed block diagram of the system.

The generated random key is then fed to the Key Scheduling module to generate different random sub-keys for each round. An explanotary block diagram for the AES with LFSR module is shown in figure 5.

B. DPR

An IP core of the AES with LFSR module explained above is used with the DPR feature of the FPGA to decrease the resources utilization. The encryption or decryption module (that performs one round of encryption or decryption) is the reconfiguration parameter. This means that only one module of the two reconfigurable modules is loaded on the FPGA at a time, the other can replace it during runtime if required. Hence, the FPGA's resources used are not the sum of the modules' resources, rather the number of resources of the bigger module.

The system is implemented with the DPR flow using Vivado 2015.2. ZC702 evaluation board Processing System (PS) and Programmable Logic (PL) partitions are used. The PS (ZYNQ7) is composed of an ARM Cortex A9 processor and a UART. While the PL has the AES with LFSR IP core, the AXI-HWICAP, and the configuration memory as shown in figure 6. UART of the PS is used to input the values serially to the AES with LFSR IP core and display the output. AXI4-Lite interface is used to connect the system, with the PS as the master, and our IP core and the HWICAP as the slaves. The PS is the master as the user chooses to reconfigure through it (using UART) and it performs the reconfiguration automatically. Also as it is used to input the values to our IP core.

A full BIT file is initially programmed on the FPGA with one default RM. If a change is requested, a partial BIT file is loaded from an SD card that stores the partial BIT files.

TABLE I
RESOURE UTILIZATION CONSUMED BY DIFFERENT ALGORITHMS.

Algorithm	Slice LUTs	Slice REGs	F7 muxes	F8 muxes
AES with LFSR with Enc and Dec Round (no DPR)	3585	2580	310	72
AES with LFSR with Encryption Round (DPR)	1949	2051	104	40
AES with LFSR with Decryption Round (DPR)	2683	1731	168	8
Burman DPR on AES encryption [5]	4369	1512	not specified	not specified

The PS fetches the targeted partial BIT file from an external memory (SD card), then send it to the AXI-HWICAP. In its turn it sends the data to the ICAP which saves the data in the FPGA's configuration memory for reconfiguration.

Now, that the system implementation is discussed, the results are presented in the following section.

IV. RESULTS

The number of clock cycles taken vary for each process. The key scheduling process takes 55 clock cycles to finish saving all the 10 sub-keys of its input key to the DRAM. While the total encryption process with LFSR consumes 129 clock cycles. For the decryption process with LFSR, it takes 78 clock cycles.

When the processor is ready, and the required partial BIT file is installed on the FPGA, the process begins. Inputs are fed to the system and outputs are displayed serially using UART, through the terminal emulator of the Xilinx SDK. The terminal interact with the UART through the port connected to the FPGA, with a baud rate of 115200K Symbols/sec. The following subsections discuss the results.

A. Resource utilization

The resources utilized are shown in table I. Using DPR decreased the Slice LUTs utilization by 45.6% when encrypting, and 25.1% when decrypting when compared with our design of both encryption and decryption modules loaded (No DPR). While the Slice REGISTERs utilization was decreased by 20.5% when encrypting, and 32.9% when decrypting. F7 multiplexers' consumption was decreased by 66.45% for encryption, and 45.8% for decryption. 44.4% and 88.8% consumption decrease of F8 multiplexers for encryption and decryption respectively. When comparing with other papers, Burman [5] used the same device as ours (XC7Z020) for implementing DPR on AES encryption. A 55% reduction in slice LUTs utilization is achieved when comparing it with our AES with LFSR with encryption module.

B. Energy utilization and throughput

When a 30 ns clock is assumed, the energy per bit for the AES with LFSR with Encryption Round is 3.6 nJ/bit, while with decryption round is 2.19 nJ/bit. Rao et al. [9] consumed 6.4 nJ/bit for encryption when using a 30 ns clock with a virtex-7 FPGA. Our algorithm has 43.75% less energy consumption for encrypting one bit when compared to [9].

The encryption throughput is 33.07 Mbps, and the decryption is 54.7 Mbps when the same clock is used. While the operational frequency of the encryption is 0.258 MHz, and of decryption is 0.427 MHz. It can be calculated using the formula below.

$$\frac{1}{no.of clock cycles taken \times clock period}$$

C. Reconfiguration time

The main drawback of the DPR is the reconfiguration time, that is the time taken to reconfigure a new RM in the specified RP other than the one currently loaded. The size of the partial bit file and the bandwidth of the configuration port (ICAP for this paper) are directly related to the speed of configuration [7]. The reconfiguration time can be approximately calculated by dividing the size of the BIT file by the throughput of the ICAP [10]. The reconfiguration time elapsed for the proposed design is 9.8775 ms for the three RMs as they all occupy the same RP having the same resources of the FPGA. The three RMs are: the encryption round, the decryption round, and blank(a module that does nothing). The reconfiguration time is relatively much longer than the encryption and decryption process, however, it is not much affecting in the iot platform as speed is not much targeted rather the area and power which were significantly decreased.

V. CONCLUSION

This paper proposed a low area and power cryptographic algorithm using AES-128. DPR feature of an FPGA was used to decrease the resources utilized. As this feature permits reconfiguring selected parts of the system while the rest of the system is still functioning. The parameter of reconfiguration was one round of encryption and one round of decryption. The system altered between encryption and decryption modes of AES-128 during runtime using DPR to save resources. The hardware resources consumed decreased by an average of 33% for encryption and 29% for decryption when comparing with the same system without using DPR. While the energy consumption was decreased by 43.75% when comparing with a conventional design. Hence, DPR can be used in the IoT platform with a cryptographic algorithm to decrease the hardware and power resources utilized by the system.

ACKNOWLEDGEMENT

This work was supported by the Egyptian Information Technology Industry Development Agency (ITIDA) under ITAC Program CFP #96 and #158.

REFERENCES

[1] Roberto Minerva, Abyi Biru, and Domenico Rotondi. Towards a definition of the internet of things (iot). *IEEE Internet Initiative*, 1:1–86, 2015.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

[4] Khaled Khatib, Mostafa Ahmed, Ahmed Kamaleldin, Mohamed Abdelghany, and Hassan Mostafa. Dynamically reconfigurable power efficient security for internet of things devices. In *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4. IEEE, 2018.

[5] Shuchishman Burman, P Rangababu, and Kamalika Datta. Development of dynamic reconfiguration implementation of aes on fpga platform. In *2017 Devices for Integrated Circuit (DevIC)*, pages 247–251. IEEE, 2017.

[6] Snehal Wankhade and Rashmi Mahajan. Dynamic partial reconfiguration implementation of aes algorithm. *International Journal of Computer Applications*, 97(3), 2014.

[7] Xilinx Inc. Vivado design suite partial reconfiguration user guide ug909.

[8] Ahmed Kamaleldin, Ahmed Mohamed, Ahmed Nagy, Youssef Gamal, Ahmed Shalash, Yehea Ismail, and Hassan Mostafa. Design guidelines for the high-speed dynamic partial reconfiguration based software defined radio implementations on xilinx zynq fpga. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[9] Muzaffar Rao, Thomas Newe, and Ian Grout. Aes implementation on xilinx fpgas suitable for fpga based wbsns. *2015 9th International Conference on Sensing Technology (ICST)*, 2015.

[10] Shady Soliman, Mohammed A Jaela, Abdelrhman M Abotaleb, Youssef Hassan, Mohamed A Abdelghany, Amr T Abdel-Hamid, Khaled N Salama, and Hassan Mostafa. Fpga implementation of dynamically reconfigurable iot security module using algorithm hopping. *Integration*, 2019.