Efficient Implementation of the AES Algorithm for Security Applications

Abstract— **Throughput , area and power optimized designs for the advanced encryption standard algorithm are proposed in this paper. The presented designs are suitable for the encrypt-only AES-128 algorithm. Both designs integrate pipelining and iterative architectures in one design. This is achieved through applying the concept of partial loop unrolling where iterations and multistage pipelining are used to optimize area , throughput and dynamic power consumption. The first design achieves a throughput of 34.08 Gbps, an operating frequency of 266.29 MHz with 521 slice registers. The second design achieves a throughput of 34.09 Gbps, 674 slice registers and a frequency of 266.33 MHz on a Xilinx 14.2 Virtex-5 XC5VLX50-3 FPGA device. Both designs consumes an approximate value of 455 mW of dynamic power using Vivado 2014.4 on Zynq-7000 XC7Z010clq225-3 FPGA device.**

Keywords— AES; round; pipelining; throughput; area; power loop unrolling.

I. Introduction

The continuous expansion in the number of internet and wireless communications users led to many security issues such as data privacy over the insecure networks to arise. Thus, cryptography became one of the main approaches to secure and overcome attacks on the user's data.

In November 2001, the National Institute of Technology and Standards (NIST) approved the AES cryptographic algorithm as the new encryption standard for its high security and flexibility[1]. Ever since, many designs were introduced that were either aiming at high throughput as[2-8] or at low memory consumption as [2]. On the other hand, some were aiming at optimization between both parameters as [3],[4] and [5] however, almost none were targeting to develop a design that consumes low power.

In this paper, area, throughput and power optimized designs for AES-128 bit algorithm are introduced. AES-128 bit is composed of 10 rounds with a key expansion module that generates 10 keys for the 10 rounds. These keys are either generated , saved and used later for the 10 rounds [10-13] or they are computed on the fly for each round [14] and [15]. The proposed designs reduce area and power by exploiting the iterative looping idea as well as the key expansion module which computes the keys on the fly. They also makes use of pipelining through multistage pipelined registers to achieve best throughput possible. Moreover, this optimization makes these designs compatible with latest security applications, especially those embedded in low power modules such as: Xbee and Bluetooth low energy (BLE) which are the main enabling technologies for internet of things (IoT) applications.

The rest of this paper is organized as follows: Section II gives a brief description for the AES algorithm. Section III presents the first AES encryption design. In Section IV, the modification for the second design is illustrated while section V discuss the power reduction approach. In section VI the implementation results are provided and compared to previously reported FPGA designs. Finally, the work is concluded in section VII.

II. AES algorithm

The Advanced Encryption Standard (AES) algorithm is a symmetric block cipher that comprises three block ciphers, AES-128, AES-192 and AES-256[9]. Each cipher encrypts and decrypts data in blocks of 128 bits using cryptographic keys of 128-, 192- and 256-bits, respectively [3]. Based on the key length used, the number of execution rounds of the algorithm is 10, 12 or 14 respectively.

The proposed design is based on the AES-128 Encryption. Each 128-data bits block along with the 128-bit cipher key are processed through a 4 x 4 state matrix and key matrix respectively. At the start of the algorithm, the state matrix is initialized with the original plain text while the key matrix is initialized with the user input key. Fig.1shows the AES-128 encryption steps that consists of 10 rounds. Through each round, Each of the two matrices proceeds in different paths undergoing different procedures and their outputs are combined at the end of each round in the *AddRoundKey* phase. The round block that processes the state matrix consists of four main transformations: *SubBytes, ShiftRows, MixColumns* and *AddRoundKey*. The algorithm starts by executing the *AddRoundKey* transformation on the original plain text and cipher key. Moreover, the last round of the algorithm differs from the previous 9 rounds as it excludes the

Mix Columns transformation. On the other hand, the cipher key undergoes different processing steps as it is subjected to a Key-Schedule operation where it expands into 10 keys throughout the 10 rounds of the algorithm. Each step is explained in the following sub-sections.

A. SubBytes Transformation

A non-linear substitution for each byte in the state matrix with a corresponding byte value based on a lookup table called S-box (Substitution Box). Each value in this table is calculated by determining the multiplicative inverse over GF(2^8) followed by an affine transformation.

B. ShiftRows Transformation

Each row of the state matrix has its bytes cyclically shifted to the left by a certain offset, Where Row n is cyclically shifted to the left by n - 1 bytes. This means that the first row is left unchanged.

C. MixColumns Transformation

A transformation in which each 4 - byte column of the state matrix is considered as a four-term polynomial over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$, where:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{0l\}x + \{02\}. \quad (1)$$

D. AddRound Key Transformation

A simple bitwise XOR operation between each byte in the state matrix with its corresponding byte in the key matrix. The key matrix corresponds to the same round of the state matrix.

E. KeyExpansion

This phase generates 10 different expanded keys from the original key cipher to be used respectively in each round of the AES algorithm. Each column in the new key is generated from its preceding expanded round key column as described by (2) and (3).

$$W_{r,i} = W_{r-1,i} \text{ XOR } W_{r,i-1}, \quad 0 < i < 4, \quad (2)$$

$$W_{r,0} = SubWord(RotWord(W_{r-1,3})) \text{ XOR } Rcon[r] \text{ XOR } W_{r,0}. \quad (3)$$

The notation *Wr,i* refers to the 4-byte column number i in the round number r, Where r is from 0 to 10. The function *SubWord* in (3) applies the *SubBytes* transformation on each of the 4 bytes of the given word. The function *RotWord* performs a left cyclic shift by 1 byte on the given word. Furthermore, the *Rcon[r]* is the 32-bit word given by [x^{r-1} , *{00}, {00}, {00}*]. The implementation of these transformations is explained in the next section.

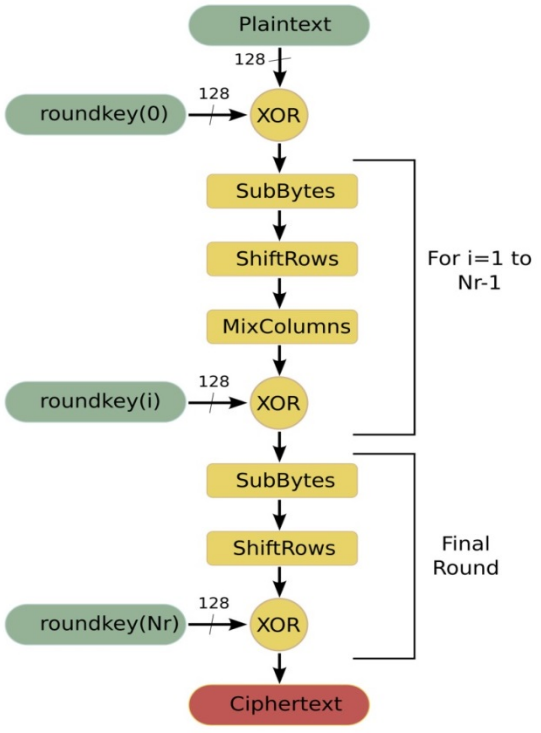


Fig. 1: AES algorithm

III. AES encryption design 1

The idea of this design is based on iterative looping integrated with pipelining rounds in order to optimize area and throughput in addition to reducing power consumption as much as possible. The hardware details of the design are shown in fig. 2. As shown in figure 2, pipe registers serve to increase throughput via pipelining data between the multi round components, the xorer component's function is a simple bitwise XOR operation between plain text and input user key, the control unit is composed of an FSM to generate proper control signals to be used in the multi round components to adjust the flow of data inside them. The details of the main components are demonstrated in the following subsections.

A. Multi-round implementation

This component serves the idea of partial loop unrolling. The hardware details are shown in fig. 3. The iteration is done 5 times before the component receives new data. This is controlled by the signals coming from the FSM control unit. The multiplexers chooses weather to pass the new data or the data coming from the previous iteration through the control signals while the 5x1 multiplexer chooses the proper round constant *Rcon[r]* based on the round order.

B. Round implementation

As described in setion II, a round consists of 4 steps: *SubBytes, ShiftRows, MixColumns* and *AddRoundKey*.

Subbytes implementation is done using 40 single port 256x8 ROMs for the 20 s-boxes of each *multi-round* component. This idea is based on the look up table of the s-box values which is much less complicated than on-the-fly computation of s-box values that requires more latency. *Mixcolumns* transformation is implemented as mentioned in section II where each byte of the state matrix is multiplied by *{1},{2}* or *{3}* as stated in (1).

Multiplication by *{3}* is implemented by left shift by 1 followed by an XOR with the initial un-shifted value, multiplication by *{2}* means left shift by 1 only and multiplication by *{1}* means no change in the value. Moreover, a conditional XOR with *{1B}* is performed after shifting if the shifted value exceeds *{FF}*.In the last round, this step is excluded through a signal generated form the FSM control unit.

C. Key generation implementation

There are 2 ways to generate round keys, they are either generated on-the-fly for each round or the whole 10 round keys are generated first then distributed over the 10 rounds. Despite the fact that the latter would consume only one component, in this design the first way was used as it doesn't add any extra delay for the critical path of the design as well as consuming 2 components which has no significant effect over the whole design. An illustration for the key expansion is shown in fig.4. The *key generation* component receives the input key, handles it column by column where the first column of the new key is generated through the same way mentioned in (3), while the remaining 3 columns are generated as in (2). The process repeats itself for each round iteration till the first 5 iterations are done, the output key is pipelined to the next stage as it undergoes the same steps in the second *key generation* component until the 5 round iterations are done and the cipher is generated.

IV. AES encryption design 2

This design has the same internal components including round, key generator and pipe register components as the first design. The hardware implementation of the design is shown in fig. 5. As shown in the figure, the main difference is the flow of data between these components. Despite the same number of key schedule and round components, the data are processed through the 2 rounds and the 2 key schedules 5 times with the pipe register in between these components before generating the cipher instead of looping 5 times through each component then passing the data to the next stage in the first design. The control unit is not composed of a FSM as in the first design, but a simple behavioral syntax that is translated into a logic that generates the proper counter signals to adjust the flow of data in a suitable order.

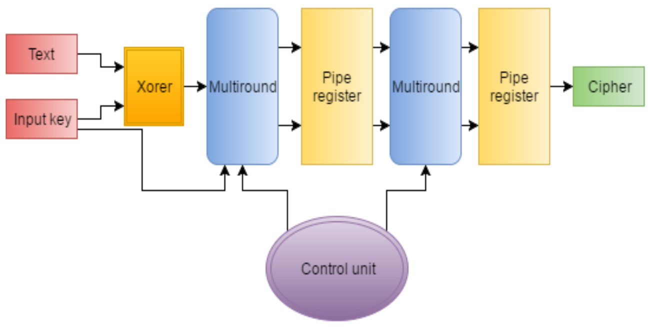


Fig.2: AES encryption design 1

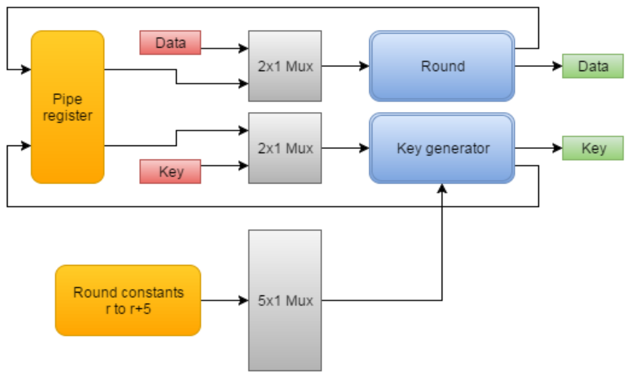


Fig. 3: Multi-round design

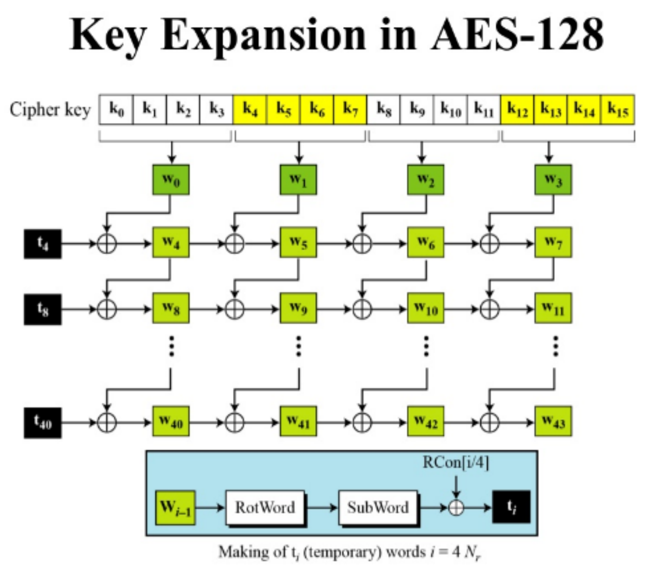


Fig.4: Key expansion mechanism

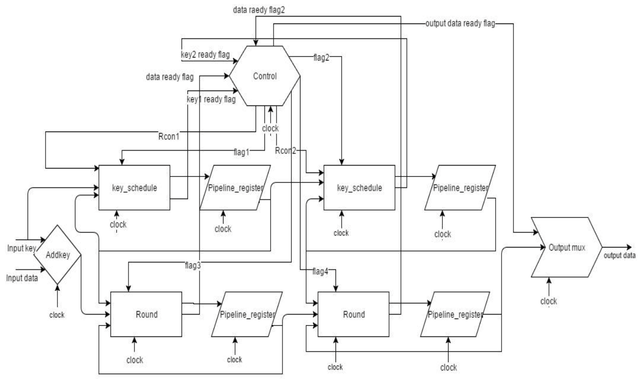


Fig.5: AES encryption design 2

V. Power analysis

There are three major strategies in FPGA power consumption reduction: The effective capacitance of resources, the resources utilization, and the switching activity of resources. The effective capacitance corresponds to the sum of parasitic effects due to interconnection wires and transistors. Since FPGA architecture usually provides more resources than required to implement a particular design, some resources are not used after chip configuration and they do not consume the dynamic power. This is referred to as resource utilization. Switching activity represents the average number of signal transitions in a clock cycle. Generally, it depends on the clock itself. The proposed design adopted the second and the third ones through using minimum number of components to reduce area, it also used the least possible amount of pipeline registers as they are the most dynamic power consuming components that consist of many signals undergoing rapid clock transitions. This power optimized design suits many IoT enabling technologies such as: Xbee and BLE modules which operate at data transmission rates in the range of Kbps or even a few Mbps. Fig.3 shows the power synthesis results discussed in the next section while table 2 shows a comparison with previous designs. It is important to note that the operating frequency of the proposed designs was degraded to 100 MHz to deliver a proper comparison with the previous designs.

VI. Synthesis results and comparisons

Both designs are implemented using VHDL and synthesized using Xilinx ISE 14.2. table 1 compares the results of the proposed designs on XC5VLX50-3 Virtex 5 device where the frequencies are 266.29 and 266.33 MHz respectively. Table 3 compares the results of the proposed designs with some previously implemented designs on the same device. As shown in table 3, the proposed designs show competitive throughput 34 Gbps and with an optimized efficiency of 65.42 and 50.58 Mbps/slice respectively, while fig.6 provides the detailed dynamic power consumption calculations and distribution over both designs. As shown in fig.6, the total dynamic power consumption is 455 mW where logic contributes to 105 mW, 19 mW for clock, 164 mW for signals and finally I/Os which consume 167 mW. Power calculations were done using Vivado 2014.4 design suite and synthesized on Zynq-7000 XC7Z010clq225-3 FPGA device.

VII. Conclusion

This paper proposed 2 AES encryption designs based on the idea of integrating between iterative looping and pipelining to optimize between area, throughput and power to provide a competitive design ready to use in IoT low power enabling technologies . These designs were implemented using VHDL and synthesized using Xilinx ISE 14.2 on XC5VLX50-3 Virtex 5 FPGA device. The results showed a competitive throughput of 34 Gbps, and efficiency of 65.42 and 50.58 Mbps/slice respectively. Both designs were also synthesized using Vivado 2014.4 design suite and mounted on Zynq-7000 XC7Z010clq225-3 FPGA device to provide dynamic power consumption calculations. The results showed that the total dynamic power consumption reached 455 mW. Which prove that the proposed designs are qualified enough for latest security applications implemented on low power modules such as Xbee and Bluetooth low energy(BLE) which are the most recommended modules for internet of things(IOT) applications.

Table 1: Synthesis results

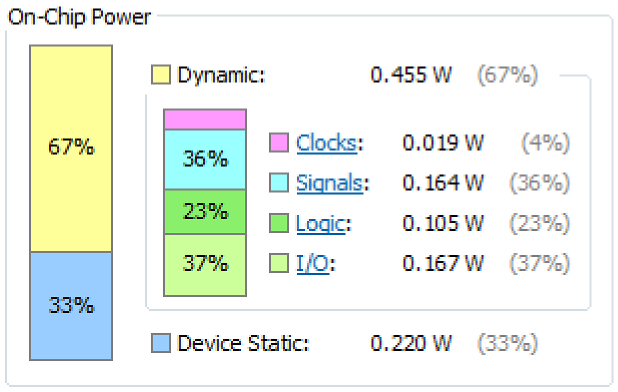
Parameter	Design 1	Design 2
Vendor	Xilinx	Xilinx
device	XC5VLX50-3	XC5VLX50-3
Slices	521	674
Slice LUTs	2490	1431
Frequency(MHz)	266.29	266.33
Throughput(Gbps)	34.08	34.09

Table 2: Dynamic power comparison with previous designs

parameter	Design 1	Design 2	[16]	[17]
Vendor	Xilinx	Xilinx	Xilinx	Xilinx
Device	Virtex 5	Virtex 5	Virtex 4	Stratix 2
Power consumption (mW)	170	170	283	301

Table 3: results comparisons with existing designs

parameter	Design 1	Design 2	[13]	[4]
Vendor	Xilinx	Xilinx	Xilinx	Xilinx
Device	XC5VLX 50-3	XC5VLX 50-3	XC5VLX 50-3	XC5VLX 50-3
Slices	521	674	303	399
Slice LUTs	2490	1431	564	1338
Frequency (MHz)	266.29	266.33	425.46	339.1
Throughput (Gbps)	34.08	34.09	1.33	4.34
Latency (ns)	45.06	45.06	96.35	29.5
Efficiency (Mbps/slice)	65.42	50.58	4.389	10.87



On-Chip Power

Dynamic:	0.455 W	(67%)
Clocks:	0.019 W	(4%)
Signals:	0.164 W	(36%)
Logic:	0.105 W	(23%)
I/O:	0.167 W	(37%)
Device Static:	0.220 W	(33%)

Fig.6: Power calculation results

References

[1] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," 2001.

[2] H. Trang and N. Loi, "An efficient FPGA implementation of the Advanced Encryption Standard," in Computing and Communication Technologies, Research, Innavation, and Visian far the Future (RIVF), 2012 IEEE RIVF Internatianal Conference on, March 2012.

[3] P. B. Ghewari, M. J. K. Patil, and A. B. Chougule, "Efficient hardware design and implementation of AES cryptosystem,"International Journal of Engineering Science and Technology, vol.2, pp. 213-219, 2010.

[4] M. H. Rais and S. Qasim, "Efficient Hardware Realization of Advanced Encryption Standard Algorithm using Virtex-5FPGA,"International Journal of Computer Science and Network Security,vol. 9, no. 9, pp. 59-63, Sep. 2009.

[5] N. A. Christy and P. Karthigaikumar, "FPGA implementation of AES algorithm using Composite Field Arithmetic," in Devices, Circuits and Systems (iCDCS), 2012 International Conference on, 2012, pp.713-717.

[6] A. Gupta, A. Ahmad, M. S. Sharif, and A. Amira, "Rapid prototyping of AES encryption for wireless communication system on FPGA," in Consumer Electronics (iSCE), 2011 IEEE 15th International Symposium on, 2011, pp. 571-575.

[7] A.-W. Luo, Q.-M. Vi, and M. Shi, "Design and implementation of area-optimized AES based on FPGA," in Business Management and Electronic Information (BMEI), 2011 International Conference on , 2011, pp. 743-746.

[8] S. K. R. S, R. Sakthivel, and P. Praneeth, "VLSI implementation of AES crypto processor for high throughput," International journal of advanced engineering sciences and technologies, vol. 6, no. 1, pp. 022-026, 2011.

[9] FIPS 197, "Advanced Encryption Standard (AES)," November 26, 2001.

[10] V. Zhang and X. Wang, "Pipelined implementation of AES encryption based on FPGA," in Information Theory and Information Security (icmS), 2010 IEEE International Conference on, Dec. 2010.

[11] J. M. Granado-Criado, M. A. Vega-Rodriguez, J. M. Sanchez-Perez, and J. A. Gomez-Pulido, "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," integration, the VLSI journal , vol. 43, pp. 72-80, 2010.

[12] K. Rahimunnisa, P. Karthigaikumar, S. Rasheed, J. Jayakumar and S. SureshKumar, "FPGA implementation of AES algorithm for high throughput using folded parallel architecture," Journal of Security and Communication Networks, vol. 5, no. 10, October 2012.

[13] M. El Maraghi, S. Hesham and M. A. Abd El Ghany, "Real-time efficient FPGA implementation of AES algorithm," in 26th Internation System on Chip Conference, pp. 203-208, September 2013.

[14] S. K. Mathew, et al. "53 Gbps native GF(24)2 composite field AESencrypt/decrypt accelerator for content-protection in 45nm highperformance microprocessors," IEEE Journal of Solid-State Circuits, vol. 46, no. 4, pp. 767-776, April 2011.

[15] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient high performance parallel hardware architectures for the AES-GCM," IEEE Transactions on Computers, vol. 61, no. 8, pp1165-1178, August 2012.

[16] G. H. Karimian, B. Rashidi, and A.farmani, " A High Speed and LowPower Image Encryption with 128-Bit AES Algorithm", International Journal of Computer and Electrical Engineering, Vol. 4, No. 3, June 2012.

[17] Bahram Rashidi and Bahman Rashidi, "FPGA Based A New Low Power and Self-Timed AES 128-bit Encryption Algorithm for Encryption Audio Signal", I. J. Computer Network and Information Security, pp. 10-20, 2013.