Journal of Engineering Science and Technology
Vol. 11, No. 12 (2016) 1678 - 1694

PERFORMANCE COMPARISON OF ADVANCED ENCRYPTION STANDARD-128 ALGORITHMS FOR WIMAX APPLICATION WITH IMPROVED POWER-THROUGHPUT

R. AHMAD[1,*], W. ISMAIL[2]

[1]Collaborative MicroElectronic Design Excellence Centre (CEDEC), Sains@USM
Level 1, Block C, No. 10 Persiaran Bukit Jambul, 11900 Penang, Malaysia
[2]Auto-ID Laboratory, School of Electrical & Electronic Engineering
Engineering Campus, 14300 Nibong Tebal, Penang, Malaysia
*Corresponding Author: rafidah.ahmad@usm.my

Abstract

Worldwide interoperability for microwave access (WiMAX) or the IEEE 802.16 standard for broadband Internet access as well as metro-wide wireless sensor network (WSN) communication, continues to gain popularity as a technology with a significant market potential. This development makes wireless security a very serious concern. Advanced encryption standard (AES) has been widely used for protection in WiMAX applications. This study focuses on the performance comparison of AES algorithms with 128-bit cipher key using field programmable gate array (FPGA) in terms of architectural, throughput differences, and power consumption rates. Different AES-128 designs, including the proposed improved power-throughput AES-128 design, are introduced and evaluated. Xilinx Virtex2 and Virtex5 FPGAs are utilized for implementation and verification purposes. Results show that with Virtex2 and Virtex5 FPGAs, the design core of the proposed AES-128 is the smallest with reduced slices up to 47%, and the throughput is increased by 90% compared with the reference designs. The proposed design also reduces total power consumption by 6% through Virtex5 FPGA.

Keywords: Worldwide interoperability, Microwave access, Advanced encryption standard, Field programmable gate array, Performance, Power.

1. Introduction

The explosive growth of the Internet and the WAN sensor communication over the last decade has led to an increased demand for high-speed and ubiquitous wireless access. One of the most popular current technologies in wireless

Nomenclatures

c	Column number
i	Number between 0 and $N_b(N_r + 1)$
N_b	Number of columns
N_k	Number of columns of the cipher key
N_r	Number of rounds
r	Row number
s	*State*
$s[r,c]$	Individual byte of the *State*
s_{rc}	Individual byte of the *State*
w	Word

Abbreviations

AES	Advanced Encryption Standard
CAM	Content Addressable Memory
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Semiconductor
DES	Data Encryption Standard
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Array
GF	Galois Field
HDL	Hardware Description Language
IOB	Input Output Boundary
LUT	Look Up Table
MAC	Medium Access Control
NIST	National Institute of Standards and Technology
OFDM	Orthogonal Frequency Division Multiplexing
PHY	Physical
ROM	Read Only Memory
VHDL	Very high speed integrated circuit Hardware Description Language
WAN	Wide Area Network
WiMAX	Worldwide Interoperability for Microwave Access
WSN	Wireless Sensor Network

communication is WiMAX, which provides high throughput broadband connections over long distances. WiMAX operates in 2.5, 3.5, and 5.8 GHz frequency bands with OFDM technology.

WiMAX has security vulnerabilities in both the PHY and MAC layers. The system is exposed to various classes of wireless attack, including interception, fabrication, modification, and reply attacks [1]. Hence, the IEEE 802.16e standard incorporated the AES algorithm to provide strong data encryption [2]. The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. Vincent Rijmen and Joan Daemen developed the AES algorithm, which is also known as the Rijndael cipher algorithm [3]. The cipher key for the AES algorithm is a sequence of 128, 192, or 256 bits known as AES-128, AES-192, and AES-256, respectively. However, this study focuses only on the AES-128 algorithm

given that large key sizes, such as AES-256, increase power and time consumption [4].

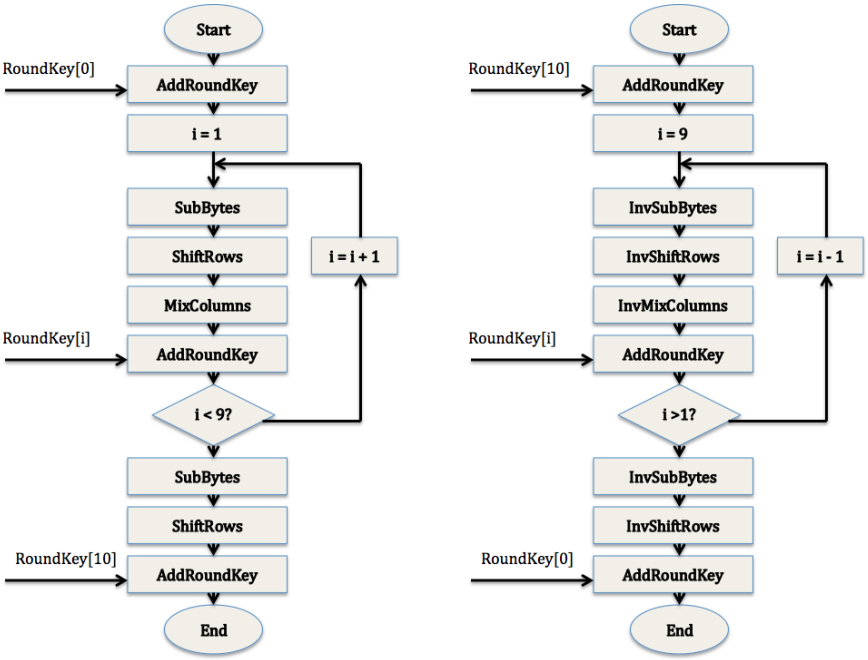
Research trends nowadays are more concerned with small high-speed security architectures and systems with low power consumption for mobile WiMAX because these devices are very compact and have limited battery power. As in the previous research works, the AES algorithm has been studied and evaluated deeply based on three areas. The first area is the architectural parameter, which is done to obtain a minimum hardware requirement that can leads to smaller design size. The second area is high throughput design, which seek to carry out encryption as fast as possible [5, 6]. Lastly, the third area is the low power design, which seeks to minimize power consumption at all costs [7, 8]. In order to achieve the best result of these three combination areas, the FPGA is used as the implementation platform. Furthermore, the reasons to implement AES algorithm in hardware are that they are faster than software implementations and intrinsically more physically secure [9].

Therefore, in this paper, an improved power-throughput of AES-128 architecture is designed with sequential technique. The proposed architecture used parallel input data and round key instead of serial implementation, in order to avoid larger input data and key size which can increase the power and time consumption. The proposed AES-128 algorithm is developed on FPGA by using Verilog HDL code. The FPGA offer math functions, embedded memories and storage elements that makes the AES design easier. Besides, the FPGA can be reconfigured for multiple tasks with only a single chip. This characteristic saves design time; thus, the product can be immediately delivered to the market. Virtex-2 Pro XC2VP100 FPGA and Virtex-5 XC5VFX70T families from Xilinx, Inc., are selected to implement the AES-128 algorithms.

This paper is organized as follows. Section 2 provides a brief overview of the AES-128 algorithm. A brief description of related AES designs from other research works is presented in Section 3. Section 4 introduces the proposed AES-128 design for performance comparison. The performance of the AES-128 designs in terms of architecture, throughput, and power consumption are compared in Section 5. The conclusion is presented in Section 6.

2. AES-128

AES is currently used for data security in WiMAX networks to replace the aging and insecure DES [2]. Although the original submission has provisions for variable length input blocks, AES takes the 128-bit plain text input and produces a 128-bit cipher text output. The number of rounds performed during the execution of the algorithm is dependent on the key size. For AES-128, the plaintext goes through N_r = 10, N_b = 4, and N_k = 4. Each encryption round performs different operations, such as byte substitution (*SubBytes*), shift rows (*ShiftRows*), mix column (*MixColumns*), and addition of a round key (*AddRoundKey*) [10]. For the decryption round, inverse transformations, such as *InvSubBytes*, *InvShiftRows*, *InvMixColumns*, and *AddRoundKey*, are necessary. The structure of the encryption and decryption rounds based on the AES standard issued by the NIST [11] is shown in Fig. 1.



(a) Encryption Round. **(b) Decryption Round.**

Fig. 1. Structure of AES-128 based on [11].

Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes known as the *State*. In the *State* array denoted by the symbol *s*, each individual byte has two indices, with its r in the range of $0 \leq r < 4$ and its c in the range of $0 \leq c < N_b$ [11]. An individual byte of the *State* is referred to as either $s_{r,c}$ or $s[r,c]$. Each of the basic transformations in Fig. 1 is briefly described below according to [11].

- *SubBytes* is a non-linear byte substitution that operates independently on each byte of the *State* using a substitution table called *S-box*. *S-box* entries are generated by computing the multiplicative inverses in Galois Field, GF(2^8), and applying an affine transformation.
- *ShiftRows* is a transformation in the cipher that processes the *State* by cyclically shifting the last three rows of the *State* by different numbers of bytes (offsets). The first row, $r = 0$, is not shifted.
- *MixColumns* operates on the *State* column by column and mixes data independent of one another to produce new columns.
- *AddRoundKey* is a transformation in the cipher and inverse cipher in which a round key is added to the *State* through an XOR operation. The length of a round key equals the size of the *State*.
- *Key Expansion* is a routine used to generate a series of round keys from the cipher key. The cipher key is generated with *SubWord*, *RotWord*, *Rcon*, and $w[i\text{-}N_k]$, where $0 \leq i < N_b$ (N_r+1).

- *SubWord* is a function that employs a four-byte input word and applies an *S-box* to each of the four bytes to produce an output word.
- *RotWord* is a function that employs a four-byte word and performs cyclic permutation.
- *Rcon* is a round constant word array.
- $w[i\text{-}N_k]$ is an earlier position of word N_k.
- *InvSubBytes* is a transformation in the inverse cipher; it is the inverse of *SubBytes*.
- *InvShiftRows* is a transformation in the inverse cipher; it is the inverse of *ShiftRows*.
- *InvMixColumns* is a transformation in the inverse cipher; it is the inverse of *MixColumns*.

3. Related Works

This section discusses existing research works on FPGA-based designs for the AES algorithm with sequential approach, ordered from the previous to the most recent research works. These works are presented to show the different possible designs developed to achieve a high throughput as well as a compact area.

Satyanarayana [12] employed a VHDL code to design an AES-128 algorithm for encryption and decryption modes. A sequential technique was implemented on this algorithm. The code was written originally with 128-bit parallel ports for both input data and key data but was then converted into 64 bits each to save I/O pins. The AES core only has three sub-modules for *S-box* transformation and key expander operations. The *S-box* values were stored in the memory to speed up the AES architecture. However, the researcher did not include the pin for inverse key expansion in the architecture. Both Virtex2 XC2VP100-FF1696 and Virtex5 XC5VFX70T-FFG1136 FPGAs have been used for data verification.

A high-throughput sequential and fully pipelined AES-128 architecture was proposed in [13]. The AES hardware blocks are composed of five operational modules which including the *SubBytes*, *ShiftRows*, *MixColumns*, *AddRoundKey*, and *Key Expansion* circuits. However, the authors only focused on the design of high performance architectures of three blocks: *SubBytes*, *MixColumns*, *Key Expansion*. The architecture is based on the CAM *SubBytes* and *InvSubBytes* schemes. With three-phase pipelining, the proposed CAM-based *SubBytes* scheme requires a smaller gate delay than that required by the ROM-based design with a Xilinx core generator. This paper also proposed a high-speed *MixColumns* and *InvMixColumns* functional blocks with a hardware-sharing structure. These blocks required 23 XOR gates and achieved smaller gate delay than the reference architectures. The *Key Expansion* circuit is comprised of 32-bit parallel multiplexers, XOR operation, and *S-box* that is similar to *SubBytes* module. The AES-128 algorithm was implemented on Xilinx Virtex2 XC2V3000 FPGA with 128-bit parallel input data and key data. The results of sequential AES showed that approximately 53% of the slices were used and a throughput of 0.876 Gbps was obtained.

Rais and Qasim conducted research on the AES-128 algorithm in [14, 15]. In both papers, AES-128 with a high-performance *S-box* was developed using the reduced residue of prime numbers. The researchers claimed that the designed *S-box* adds more confusion to the entire process of the AES algorithm; the algorithm is then made more complex and provides further resistance against attacks. At the best of our knowledge, the structure of the proposed AES algorithm was designed sequentially. The AES architecture is based on 128-bit version with 10 rounds, and 4x4 matrix *State* bytes. The architecture was developed with VHDL code on Virtex5 XC5VLX50-FFG676 FPGA. The results of this study showed that the slices used are 24% and throughput was 3.090 Gbps.

Another AES-128 algorithm with efficient area-throughput was designed using VDHL in [16]. In the proposed design, an iterative architecture is chosen to suit the entire design in a single FPGA chip. This approach has several variations including multiple copies of an iterative implementation of parallel processing, a partially pipelined implementation, and a combination of hybrids. The iterative approach was done followed by efficient utilization of a number of operations in each clock cycles. In order to obtain the small silicon area and high throughput, three modifications on the AES architecture were proposed: merger of *SubBytes* and *ShiftRows*, generating the *SubBytes* for encryption and employs a LUT to store the coefficients of the inverse *S-box* for decryption, and each clock cycle is assigned to complete a set of operations. With the *S-box* stored in a ROM/RAM-based LUT, the issue of *S-box* dominates the hardware complexity could be avoid. Xilinx Virtex2 XC2VP30-FF896 FPGA was utilized to verify the algorithm. The implementation results obtained a throughput of 1,403 Mbps and 45% slice usage. The speed also improved four times compared to the original AES design.

The high throughput and low area AES-128 algorithm proposed by Litochevski and Dongjun [17] was analyzed in this paper. The algorithm was designed with Verilog, which comprises one sub-module for key expansion and five sub-modules for encryption and decryption modes. All these sub-modules work sequentially in two modes of operation, which can be selected on the fly using an input signal. The AES core interfaces are implemented with full width of the required data to minimize the time required to load the information. In order to support high data rates, the core enables pipeline operation of four vectors simultaneously. However, the key output pin for the inverse key expansion based on the AES standard [11] was not included in the AES design. This key is important in cipher transformations, which can be inverted and then implemented in the reverse order to produce a straightforward inverse cipher for the AES algorithm. This architecture employs parallel 128-bit input data and key data. Both Virtex2 XC2VP100-FF1696 and Virtex5 XC5VFX70T-FFG1136 FPGAs also have been used for data verification.

The summary of the analysis on the AES-128 implementation results obtained by [12-17] for both encryption and decryption modes are shown in Tables 1 and 2 using Virtex2 and Virtex5 FPGAs, respectively. These FPGA families were selected in order to see how the proposed architecture and reference architectures functioned in different hardware technology that would affect the design core, throughput and power consumption.

Tables 1 and 2 show that the best AES-128 architecture from previous research works is in [12] which requires only 1% of the slices of Virtex5 FPGA and highest throughput per slice at 1.870 Mbps/slice. The comparison of throughput per slice is the most objective manner of comparing different AES-128 architectures depending on the size and transmission speed. However, the throughput achieved by [12] using Virtex2 FPGA is only 0.832 Gbps, which is the lowest among others. A low throughput indicates a slow timing process and high power consumption. The highest throughput of 3.090 Gbps in [14, 15] used 24% slices of Virtex5 family; however, the throughput per slice obtained is 5% less than that obtained in [12]. The largest AES-128 core used is 53% slices of Virtex2, which is designed by [13]. Architecture in [17] achieved the lowest power consumption for both Virtex2 and Virtex5.

These implementation results prove that in reality, achieving a small design core with high speed and low power consumption at the same time is difficult. Hence, this paper aims to achieve the best performance of the AES-128 algorithm compared to previous works by using the least amount of hardware and power possible, through sequential approach. From the studies presented above, FPGAs are considered one of the most important hardware platforms and integral part for the security algorithms implementation, which offer much easier and reasonably cheap solution for real time verification [18].

Table 1. Comparison of the Implementation Results Obtained from Previous Research Works on AES-128 with Sequential Approach on Virtex2 FPGA.

Ref.	Slices Used	Freq. (MHz)	Late-ncy (cycles)	Throu-ghput (Gbps)	Throughput /Slice (Mbps/slice)	Power (mW)
[12]	7806/44096 (17%)	84.5	13	0.832	0.107	646
[13]	7617/14336 (53%)	75.3	11	0.876	0.115	-
[16]	6211/13696 (45%)	142.5	13	1.403	0.226	-
[17]	3119/44096 (7%)	172.3	19	1.160	0.372	450

Table 2. Comparison of the Implementation Results Obtained from Previous Research Works on AES-128 with Sequential Approach on Virtex5 FPGA.

Ref.	Slices Used	Freq. (MHz)	Late-ncy (cycles)	Throu-ghput (Gbps)	Throughput /Slice (Mbps/slice)	Power (mW)
[12]	789/44800 (1%)	149.9	13	1.476	1.870	1473
[14, 15]	1745/7200 (24%)	242.2	-	3.090	1.771	-
[17]	1533/44800 (3%)	261.8	38	0.882	0.575	1454

4. Proposed AES-128 Design

An improved power-throughput of AES-128 architecture with parallel I/O data is proposed in the present work. The *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey* transformations were designed with Verilog and executed sequentially. The *S-box* values of totally 128 bits were stored in ROM-based LUTs in order to decrease the required gates and speed up the execution time. Part of *S-box* schematic using LUTs is shown in Fig. 2.

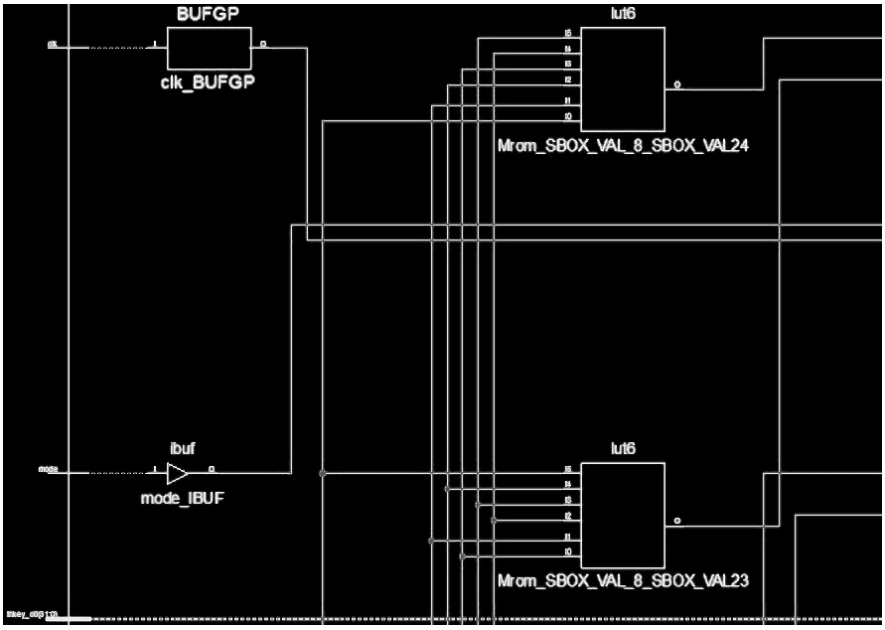


Fig. 2. Part of *S-box* in the Proposed AES-128 Core through Xilinx Software.

The proposed AES-128 design can operate both in encryption and decryption modes by selecting the control pin. The block diagram of the proposed AES-128 algorithm is shown in Fig. 3. The cipher maintains an internal, 4x4 matrix of bytes referred to as *State*, on which the operations are performed. The *State* was filled with the input data and XOR-ed with the round key. The length of a round key is 128-bit which is equal to the size of *State*. The *ShiftRows* was performed by reordering the *State* bytes while they were shifted through the unit, except for the remaining four bytes of *State*. These four bytes were processed and maintained in the other data registers of the AES-128 architecture. Then, the *MixColumns* processed the bytes of *State* column by column and mixed the data independently to produce new columns.

Similar to [17], the proposed AES-128 architecture also requires many I/O pins because many bus widths are used. Therefore, the IOBs of FPGA usage are affected. Unlike the architectures in previous research works, the proposed architecture in this work includes the key output pin for the next round of *AddRoundKey* transformation and inverse key expansion operation (as shown by the shaded block in Fig. 3). The key expansion block took in the initial key or

previous round key and organized the byte into a *State*. Each of the data blocks in the final column was shifted and the topmost block was combines with a round constant. The final column then processed as substitution transformation and the columns were added together to generate a new 128-bit round key. Table 3 shows the I/O pins available and their functions.

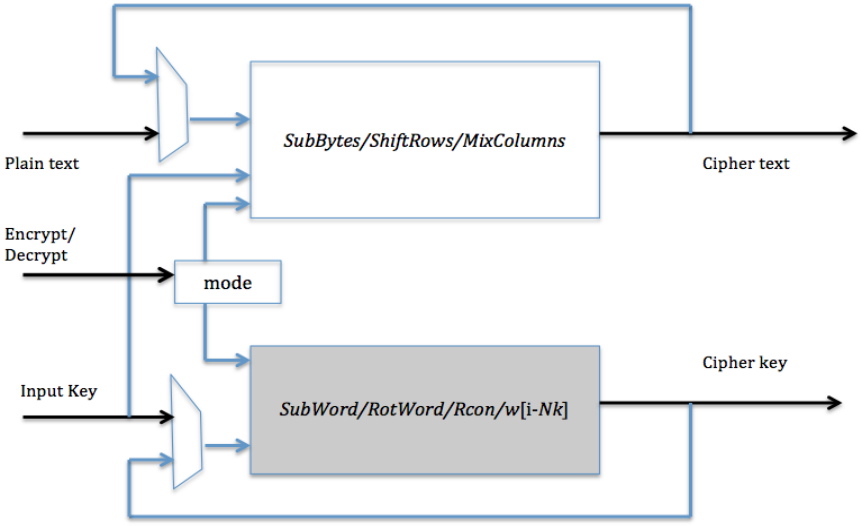


Fig. 3. Block Diagram of the Proposed AES-128 Core.

Table 3. I/O Functional Description of the Proposed AES-128 Algorithm.

Pin Name	I/O	Width	Description
clk	I	1	Global clock signal.
reset	I	1	Global core reset, active high.
count	I	4	Indicates the number of rounds, *Nr*.
mode	I	1	Encryption/decryption modes selection control.
inkey_c0, inkey_c1, inkey_c2, inkey_c3	I	32	Key value input bus for 128 bits.
s00, s01, s02, s03, s10, s11, s12, s13, s20, s21, s22, s23, s30, s31, s32, s33	I	8	State array plain or cipher text input data for a total of 128 bits.
outkey_c0, outkey_c1, outkey_c2, outkey_c3	O	32	Key value output bus for 128 bits.
s00_enc, s01_enc, s02_enc, s03_enc, s10_enc, s11_enc, s12_enc, s13_enc, s20_enc, s21_enc, s22_enc, s23_enc, s30_enc, s31_enc, s32_enc, s33_enc	O	8	State array cipher or plain text output data for a total of 128 bits.

5. Results Comparison

The performance of the proposed AES-128 architecture and the architectures from previous research works is compared in this section. This section is divided into three parts. The first part shows the architecture/hardware comparison. The second part presents the parameters of the performance comparison. The third part provides a comparison of the power consumption of the available architectures. The proposed architecture is verified and implemented on two different FPGA families, namely, Xilinx Virtex2 XC2VP100-FF1696 and Virtex5 XC5VFX70T-FFG1136 platforms.

Virtex2 is built on 0.13 µm CMOS technology process. Combining a wide variety of flexible features and IP cores, Virtex2 enhances programmable logic design capabilities and is a powerful alternative to mask-programmed gate arrays [19]. This family also incorporates multi-gigabit transceivers and power PC CPU blocks in its architecture. Virtex2 has 99,216 logic cells, 44,096 slices, and 1,164 I/O pads. Built on a 65 nm state of the art copper process technology, Virtex5 offer the best solution for addressing the needs of high performance logic, DSP, and embedded system designers with unprecedented logic, DSP, hard/soft microprocessor, and connectivity capabilities [20]. Virtex5 contains of power PC 440 microprocessor embedded blocks with 11,200 slices and 640 I/O pads [20]. Virtex2 and Virtex5 were operated at a clock frequency of 25 MHz and 50 MHz during the simulations, respectively. These AES-128 architectures could perform at these rates and would show their power requirements at near maximum performance. The implementation results of these architectures were compared to determine which of them provides better performance with high throughput, less hardware requirements, and less power consumption on different FPGA platforms.

5.1. Architectural comparison

The structure of modern FPGAs comprises a two-dimensional array of CLBs interconnected via horizontal and vertical routing channels. The CLBs typically comprise LUTs and flip flops organized in slices, which means that those elements shares connections in order to utilize fast carry chain. LUTs may be configured as either combinational logic or memory elements [21].

Table 4 shows that the proposed AES-128 architecture requires more hardware resources of Virtex2, with 6% of the slices used. This means that the architecture required a lot of logic cells, which is due to the increased of registers needed. A logic cell in Virtex2 consists of a LUT, a flip-flop, and connection to adjacent cells. The LUT uses combinatorial logic to implement a 4-input expression such as AND, OR, NAND, ADD. A logic slice consists of 2 logic cells, and a CLB consists of 4 slices. The increased slice requirement will automatically affect the throughput per slice metric as explained in Section 5.2. The IOBs usage is approximately 44% because the input and output pins use bus widths. Nevertheless, this issue does not make any impact on the total throughput of the proposed architecture. Based on Fig. 4, seems that the proposed AES-128 is still using the lowest slices of Virtex2 FPGA if compared to the reference designs. This proves that with sequential execution through Verilog approach, the registers required could be minimized.

However, the hardware requirements are different for Virtex5 FPGA as shown in Table 4. For Virtex5, the definition of a logic slice and block is not similar with Virtex2. The Virtex5 has 2 slices per CLB and has four 6-input LUTs. Additionally, the Virtex5 has 4 flip-flops per slice. The proposed AES-128 exhibits the lowest slice usage of 520 (1%) compared with the reference designs. The slice FFs and LUTs required only 4% each. From this total, the LUTs for *S-box* took about 2%. The current area-optimized algorithm of the proposed AES-128 is mainly based on the realization of *S-box* LUTs and the minimizing of internal registers, which could lessen the design core significantly. It can be concluded that from Fig. 4, the design core of the proposed AES-128 is the smallest with reduced slices up to 47% through Virtex2 if compared to the reference designs.

Table 4. Architectural Characteristics of the Proposed AES-128 Design.

FPGA Family	Slices	Slice Flip Flops	LUTs	IOBs	Global CLK
Virtex2	2954/44096 (6%)	628/88192 (0%)	5703/88192 (6%)	519/1164 (44%)	1/16 (6%)
Virtex5	520/44800 (1%)	1864/44800 (4%)	1964/44800 (4%)	519/640 (81%)	1/32 (3%)

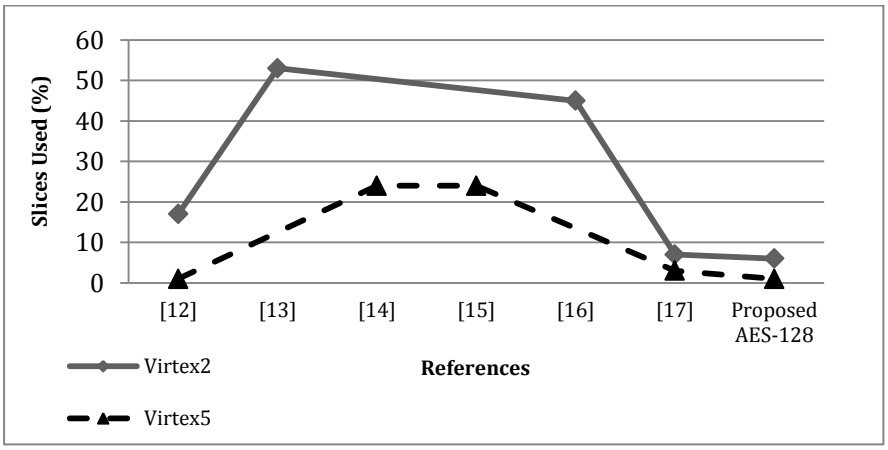


Fig. 4. Slices Used vs. References with Virtex2 and Virtex5 FPGAs.

5.2. Performance comparison

Generally, throughput is defined as the average rate of successful message delivery over a communication channel. In this paper, throughput is directed toward evaluating each architecture's characteristic and performance. Throughput is calculated as Eq. (1) based on [21]. Latency is calculated as Eq. (2).

$$\text{Throughput (Gbps)} = \frac{128 \text{ bits} * \text{Maximum Frequency (MHz)}}{\text{Latency}} \quad (1)$$

$$\text{Latency} = \text{No. of clk cycle} * N_r \quad (2)$$

If the AES-128 algorithm is directed toward a device that wakes up, captures data, encrypts data, transmits data, and reverts to sleep mode, latency can become an issue because the longer the system needs to be awake, the more power is required [22]. Hence, latency should be as small as possible to achieve a power saving system. Furthermore, longer battery lifetime is necessary, particularly for WiMAX mobile devices.

The comparison of throughput per slice can be determined with the size and transmission speed; this procedure is the most objective method of comparing different AES-128 architectures. The number of slices required and the maximum operating frequency for each implementation were obtained from the Xilinx synthesis report. The equation for throughput per slice is shown as below.

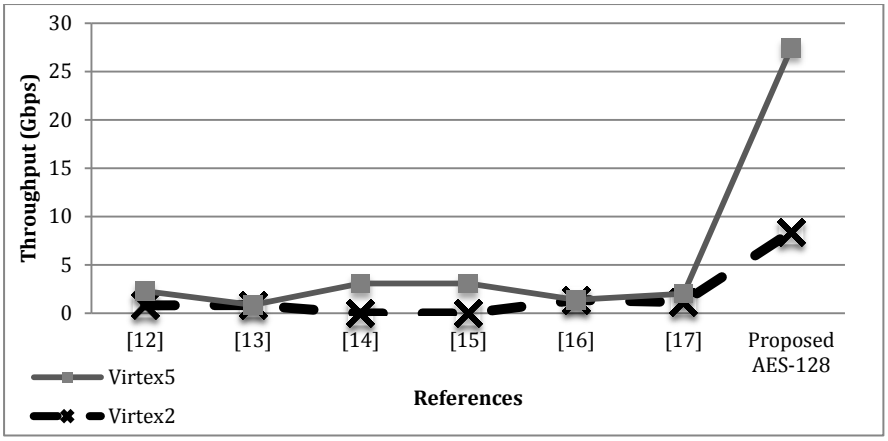
$$\text{Throughput/slice} = \frac{\text{Throughput (Gbps)}}{\text{No. of slices used}} \quad (3)$$

To compare implementations using the throughput and throughput/slice metrics, the AES architectures must be implemented on the same FPGA as shown in Table 5. Different FPGAs within the same family yield different timing results as a function of available logic and routing resources, both of which change based on the die size of FPGA [21]. This result affects the measured throughput and throughput/slice. Table 5 proves that the proposed AES-128 as an optimal implementation has the smallest latency and the highest maximum frequency, throughput, and throughput per slice. The throughput for the proposed architecture shown in Fig. 5(a) increased up to 90% for Virtex2, and 95% for Virtex5. Fig. 5(b) shows that the throughput per slice increased up to 96% for Virtex2, and 98% for Virtex5 compared with other architectures in the previous works. The highest throughput achieved by the proposed architecture indicates that the said architecture has the highest encryption speed and the best performance. Furthermore, Virtex5 shows that with a clock frequency of 50 MHz, the throughput of the proposed architecture is pushed to the maximum rate.

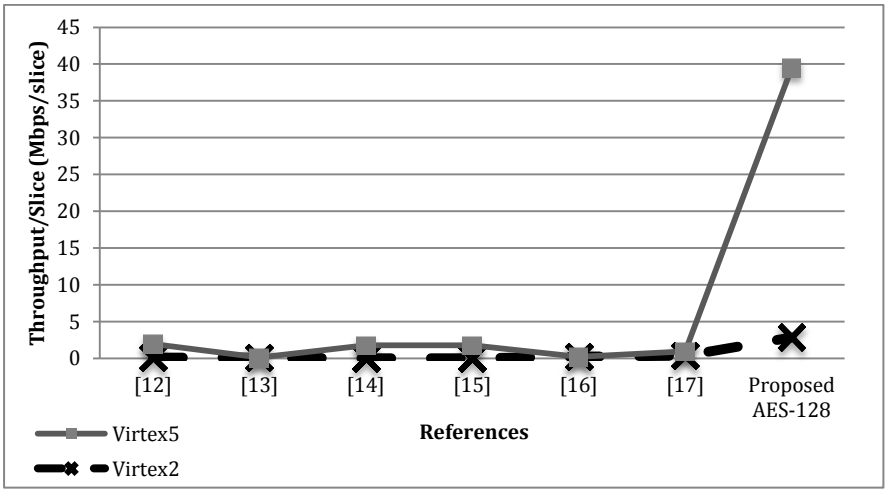
Table 5. Performance Data of the Proposed AES-128 Design.

FPGA Family	Min. clk period (ns)	Max. Freq. (MHz)	Latency (cycle)	Throughput (Gbps)	Throughput /Slice (Mbps/slice)
Virtex2	3.799	263.246	4	8.424	2.851
Virtex5	2.244	445.679	3	19.016	36.569

Meanwhile, Fig. 6 shows the list of 128-bit input data and output data from both *State* array and round key are matched to each other during encryption and decryption modes. The encryption mode started at 0.21 μs with mode '0' and the decryption mode started at 2.21 μs with mode '1'. The latency obtained is 3 clock cycles (2.27 – 2.21 μs) through Virtex5 FPGA as shown in Fig. 6(b). One clock cycle is 20 ns.



(a) Throughput vs. References

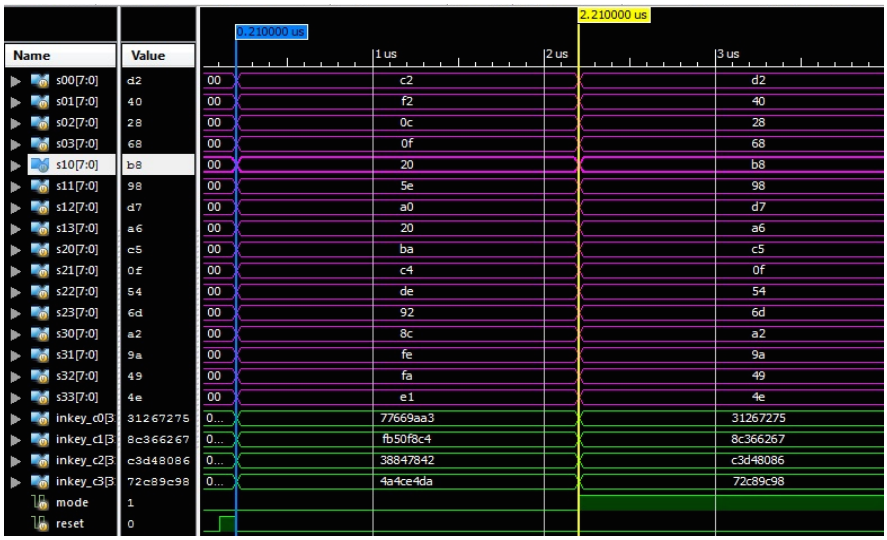


(b) Throughput/Slice vs. References

Fig. 5. Performance Comparison with Virtex2 and Virtex5 FPGAs.

5.3. Power comparison

The power requirements for the three AES-128 architectures are discussed in this section. The Xilinx XPower analysis tool was used to analyze the power consumption. The power consumed by the architectures was compared in terms of clock trees, logic, signal, and quiescent power. This analysis introduces a very efficient method of locating the blocks or parts of the design that are the most deprived in the power aspect, thereby providing an easy path to power optimization.



(a) Encryption mode

(b) Decryption mode

Fig. 6. Simulation Waveform of the Proposed AES-128.

Table 6 shows the comparison of power consumption performed by the proposed AES-128 with Virtex2 and Virtex5 FPGAs. The lowest power consumed is totally 307 mW through Virtex2. The proposed architecture requires up to 82% less logic power consumption and up to 81% less signal power consumption, if compared to [12] and [17]. The quiescent power is 204 mW for Virtex2 and 1,343 mW for Virtex5. The quiescent device power consumption is

the amount of power consumed by the FPGA prior to the device being programmed. This definition implies that the FPGA is in a non-programmed state but has been powered nevertheless.

Table 6. Power Consumed by the Proposed AES-128 Design.

FPGA Family	Power (mW)				
	Clock	Logic	Signal	Quiescent	Total
Virtex2	26	18	59	204	307
Virtex5	25	6	10	1343	1384

Figure 7 shows that the total power consumed by the proposed architecture is reduced by 6% minimally, compared to [12] and [17]. The proposed AES-128 architecture also has the fewest hardware requirements and the highest throughput through both FPGAs. With these characteristics, the proposed architecture can be implemented in WiMAX mobile devices because of the reduced design area and battery usage but high transmission speed.

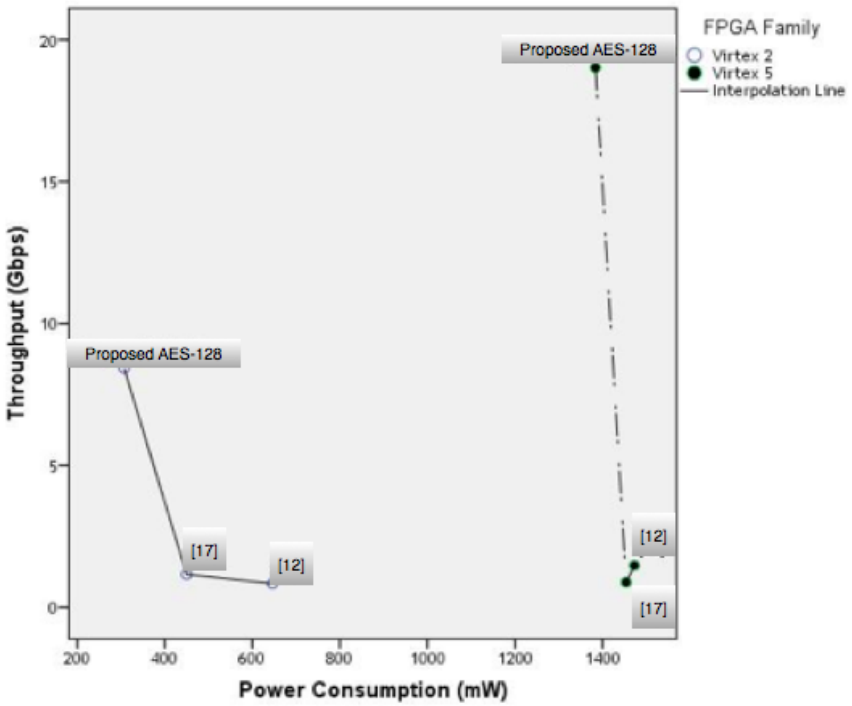


Fig. 7. Comparison of Throughput vs. Power Consumption through Virtex2 and Virtex5 FPGAs.

6. Conclusion

An improved power-throughput AES-128 design is proposed in this paper for encryption and decryption modes. The architectural and throughput differences as

well as the power consumed by the proposed and reference designs were compared. This comparison indicated that the proposed design exhibits the best performance through either Virtex2 or Virtex5 FPGA. The best performance was defined strictly as the fewest hardware requirements, highest throughput, and lowest power consumption. In overall, the findings indicate that through Virtex5 and Virtex2 FPGAs can provide the smallest design area, the highest throughput, and the lowest power consumption for the proposed AES-128 algorithm. Based on this finding, it is also be stressed that FPGA must be selected for how it best fits the selected architecture of the AES-128 algorithm because most of the power consumed in the analysis is due to quiescent power. These characteristics are important for current research trends given that WiMAX mobile devices are very compact and have limited battery power. With the improved power-throughput of the proposed architecture shown in this paper, the battery lifecycle can be expanded and this will lead to a lower cost maintenance of WiMAX devices. For future work, the proposed AES-128 algorithm will be implemented on software defined radio in order to verify its performance when transmitting and receiving data in real time.

Acknowledgment

This research was financially supported by the MOSTI ScienceFund Research Grant, Project No. 06-01-05-SF0640.

References

1. Jha, R.K.; and Dalal, U.D. (2010). A journey on WiMAX and its security issues. *International Journal of Computer and Information Technologies (IJCSIT)*, 1(4), 256-263.
2. Mishra, A.; and Glore, N. (2008). Privacy and security in WiMAX networks. *WiMAX standards and security* (1st ed.). USA: CRC Press, 205-228.
3. Daemen, J.; and Rijmen, V. (2002). AES-The advance encryption standard. *The design of Rijndael* (1st ed.). Berlin, New York: Springer-Verlag Berlin Heidelberg, 1-8.
4. Ahmad, R.; and Ismail, W. (2013). A survey of high performance cryptography algorithms for WiMAX applications using SDR. *Self-organization and green applications in cognitive radio networks* (1st ed.). USA: IGI-Global, 231-246.
5. Rodriguez, F.; Saqib, N.A.; and Diaz-Perez, A. (2003). 4.2 Gbit/s single-chip FPGA implementation of AES algorithm. *IEE Electronic Letters*, 39(15): 1115-1116.
6. Hodjat, A.; and Verbauwhede, I. (2004). A 21.54 Gbits/s fully pipelined AES processor on FPGA. *Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines*. California, USA, 308-309.
7. Chodoweic, P.; and Gaj, K. (2003). Very compact FPGA implementation of the AES algorithm. *Proceedings of the Fifth International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2003), LNCS 2779*. Cologne, Germany, 319-333.

8. Kaps, J.P.; Gaubatz, G.; and Sunar, B. (2007). Cryptography on a spec of dust. *IEEE Computer*, 40(2), 38-44.
9. Karthigai Kumar, P.; and Baskaran, K. (2010). An ASIC implementation of low power and high throughput Blowfish crypto algorithm. *Microelectronics Journal*, 41(6), 347-355.
10. Dakua, P.K.; Pradhan, M.; and Polamuri, S.R. (2011). Hardware implementation of mix column step in AES. *Special Issue of International Journal of Computer Applications on Communication and Networks*, 2(1), 6-9.
11. National Institute of Standards and Technology (NIST) (2001). Announcing the advanced encryption standard (AES). *Federal Information Processing Standards (FIPS) Publication 197*.
12. Satyanarayana, H. (2004). AES128. Retrieved April 4, 2012, from http://www.opencores.org/projects.
13. Fan, C.P.; and Hwang, J.K. (2008). FPGA implementations of high throughput sequential and fully pipelined AES algorithm. *International Journal of Electrical Engineering*, 15(6), 447-455.
14. Rais, M.H.; and Qasim, S.M. (2009). A novel FPGA implementation of AES-128 using reduced residue of prime numbers based S-box. *International Journal of Computer Science and Network Security*, 9(9), 305-309.
15. Rais, M.H.; and Qasim, S.M. (2009). FPGA implementation of Rijndael algorithm using reduced residue of prime numbers. *Proceedings of 4th IEEE International Design and Test Workshop (IDT09)*. Riyadh, Saudi Arabia, 1-4.
16. Jyrwa, B.; and Paily, R. (2009). An area-throughput efficient FPGA implementation of block cipher AES algorithm. *Proceedings of International Conference on Advances in Computing, Control, and Telecommunication Technologies*, IEEE Computer Society. Kerala, India, 328-332.
17. Litochevski, M.; and Dongjun, L. (2012). High throughput and low area AES: Core specifications. Retrieved September 2, 2012, from http://www.opencores.org/projects.
18. Rais, M.H.; and Qasim, S.M. (2009). Efficient hardware realization of advanced encryption standard algorithm using Virtex-5 FPGA. *International Journal of Computer Science and Network Security*, 9(9), 59-63.
19. Xilinx, Inc. (2011). Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet. *Product Specification*, DS083(v5.0), 1-2.
20. Xilinx, Inc. (2009) Virtex-5 family overview. *Product Specification*, DS100(v5.0), 1-2.
21. Elbirt, A.J.; Yip, W.; Chetwynd, B.; and Paar, C. (2000). An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists. *Proceedings of the Third Advanced Encryption Standard Candidate Conference*, National Institute of Standards and Technology (NIST). New York, USA, 13-27.
22. Dyken, J.; and Delgado-Frias, J.G. (2010). FPGA schemes for minimizing the power-throughput trade-off in executing the Advanced Encryption Standard algorithm. *Journal of Systems Architecture*, 56(2-3), 116-123.