FPGA IMPLEMENTATION OF AES ENCRYPTION AND DECRYPTION ALGORITHM

Aswin Gunavelu Mohan
Electrical Engineering, ECEE
Arizona State University
Tempe, AZ, USA
agunavel@asu.edu

Sanjana Ismail
Computer Engineering, ECEE
Arizona State University
Tempe, AZ, USA
sismail1@asu.edu

Mounika Oruganti
Computer Engineering, ECEE
Arizona State University
Tempe, AZ, USA
morugant@asu.edu

Abstract-**Cryptography is widely used in networks. It is highly important for data confidentiality. Advanced Encryption Standard (AES), is a specification for the encryption of electronic data approved worldwide. AES can be programmed in software or built with hardware. This algorithm enhances security with more randomization in secret keys, which results in algorithms taking up enormous memory spaces and large execution time on hardware platform. Field Programmable Gate Arrays (FPGAs), provide one of the major alternative in hardware platform scenario due to its reconfiguration nature and faster time-to-market. In this paper we demonstrate the design and implementation of a 128-bit Advanced Encryption Standard (AES) both symmetric key encryption and decryption algorithm by developing suitable hardware and by using various hardware optimization techniques to achieve a higher throughput and low latency compared to the software counterpart. Hardware evaluations are done in Virtex 7 and Virtex Ultra scale.**

Keywords: - AES, FPGA, Xilinx Virtex 7, Virtex Ultra scale

I. INTRODUCTION

The Advanced Encryption Standard (AES) is the Federal Information Processing Standard for symmetric encryption. It is adopted as an encryption standard by US government and became effective on May 26, 2002. AES is widely believed to be secure and efficient, and is therefore broadly accepted as the standard for both government and industry applications. With the rapid development of computers and communication technology a lot of sensitive information exchange with the public communication facilities. Secure and efficient transmission of information is important. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

In this paper, we describe the Rijndael's AES encryption algorithm in the symmetric key encryption. And we analyze the performance of AES on FPGA and ASIC.

II . AES ENCRYPTION AND DECRYPTION

A. *Need for Hardware Accelerator*

In general networking applications and communication systems, speed is very crucial. In such applications, cryptography is inevitable. Hence, the throughput of encryption and decryption process becomes a bottleneck in such applications. Improvement in throughput of cryptographic process can increase the efficiency of the entire process in a drastic way. Software implementation of the algorithm has limited speed even when implemented in high speed processors. In Intel's 1.7 GHz Pentium M processor, the throughput is observed to be around "480 Mbits/sec" [1] (Table 1). In Intel Core i3/i5/i7 and AMD APU with advanced exclusive AES – NI instruction set extensions, the throughput obtained is "5.4 Gbits/sec" [1]. These throughput values are sufficient for 1 or 100 Gigabit Ethernet cables, but when it comes to network servers and commonly used 100 Gigabit Ethernet cables (Table 2), these numbers are not sufficient and can slow down the entire communication system. Thus hardware implementation of the algorithm can provide higher throughput and efficiency for the specific applications. Field Programmable Gate Array (FPGA) is an integrated circuit that can be bought off the shelf and reconfigured by designers themselves. With each reconfiguration, which takes only a fraction of a second, an integrated circuit can perform a completely different function [2]. The implementation of the AES algorithm based on FPGA devices has several advantages like shorter design cycle, lower cost of the computer-aided design tools, improved verification and testing, potential for faster systems, higher accuracy of comparison. FPGA designs are compared based on very accurate post-layout simulations and experimental testing. Time and cost for developing an FPGA implementation of a given algorithm are much lower than for an ASIC implementation [3].

TABLE 1
THROUGHPUT VALUES IN DIFFERENT PLATFORMS

Platforms	Throughput
1.7 GHz Pentium M	480 Mbits/sec
Intel Core i3/i5/i7 and AMD APU [AES-NI instruction set extensions]	5.4 Gbits/sec

TABLE 2
THROUGHPUT FOR DIFFERENT TECHNOLOGIES

Technologies	Throughput
Gigabit Ethernet	1 Gbits/sec
USB 3.0	5 Gbits/sec
10 Gigabit Ethernet, USB 3.1	10 Gbits/sec
100 Gigabit Ethernet	100 Gbits/sec

B. Design overview of AES

The algorithm is comprised of three major sections: Cipher, Inverse Cipher and Key Expansion. Cipher converts data, commonly known as plaintext, to encrypted form called cipher. Key Expansion generates keys through the key scheduling process. Cipher and Inverse Cipher are composed of specific number of rounds. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key length [1].

AES operates on plain text of 128 bytes and key is 128,192,256 bits in length respectively. In this paper we use 128 bits key for explaining the process flow and for analysis of performance. An outline of the entire process is given in Figure 1. The encryption process consists of 10 rounds. There are four major types of transformations that are used in encryption process. They are:
i) SubBytes
ii) ShiftRows
iii) MixedColumns
iv) AddRoundKey

The last round of the encryption alone is different in a way that the mixed column operation will not be carried out. A 128-bit data block is divided into 8 bytes leading to byte wise operation and these 8 bytes are mapped as 4X4 matrix and each entry are called as states [4].

The algorithm is called as symmetric key process as it uses the same key for both encryption and decryption. The decryption is symmetrically opposite to that of encryption which uses the final key and the cipher text to give the plain text as output (Figure 2).

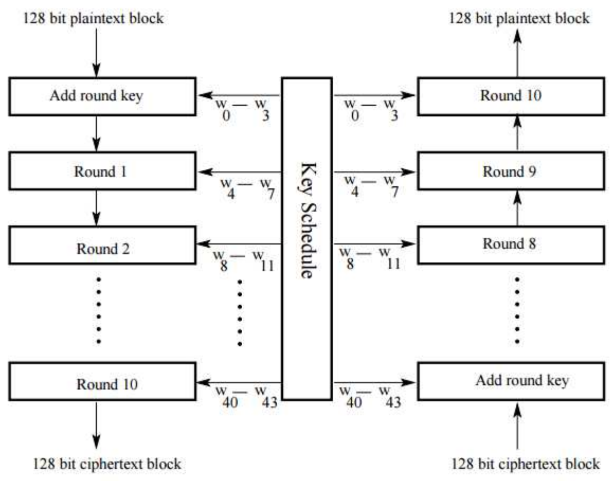

Figure 1: AES Encryption and Decryption process flow [6]

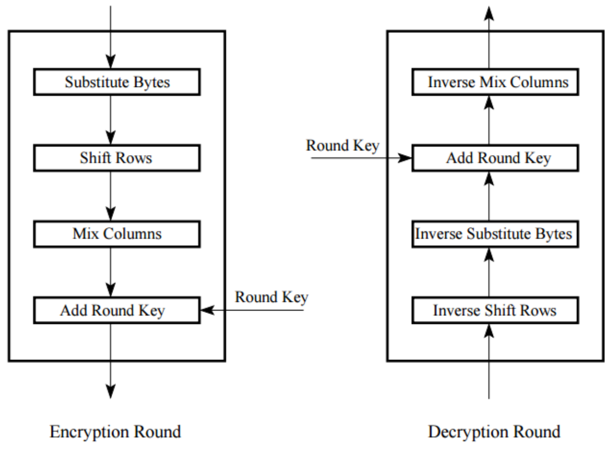

Figure 2. Transformations used in each round of AES [6]

1) SubBytes Transformation

The SubBytes transformation is a non-linear byte substitution, operating on each of the state bytes independently using a substitution table (S-box) [4]. This S-box which is invertible is constructed by composing two transformations:
1. Take the multiplicative inverse in the finite field GF (2^8), the element {00} is mapped to itself.
2. Apply the affine transformation over GF (2).
InvSubBytes is used in decryption process which uses an Inverse S-box. Process is inverse of SubBytes - firstly, inverse affine transformation of each byte element of the state matrix in the finite field G (2^8), and then calculate it's multiplicative inverse in the finite field G (2^8).

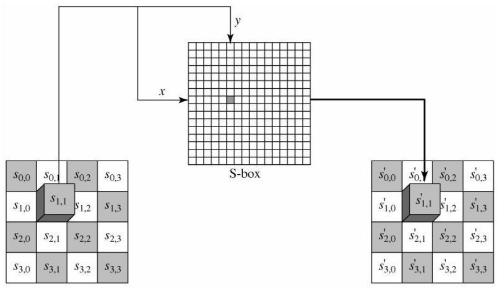

Figure 3.SubBytes transformation [5]

2) ShiftRows

In shiftrows operation the first row is not shifted. The second row is left shifted by 1 byte, the third row by 2 bytes and the fourth row by 3 bytes. Shift rows results in cyclic shifting over every row by increasing number of bytes except the first row. The inverse shift row operation used in decryption performs the same order of shifting to right.

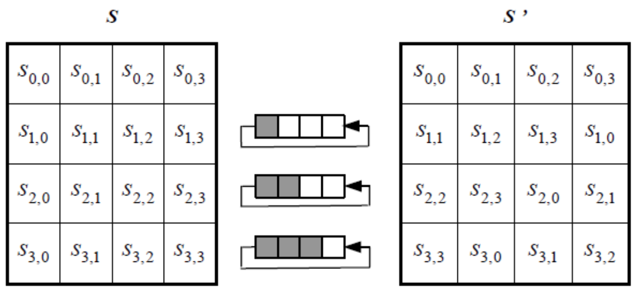
S
S '
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Figure 4. ShiftRows [8]

3) *MixColumns*

It involves the substitution of every byte of element using values over a finite field GF (2^8). Mix columns transforms each column. After the operation we get each element of the column as a linear combination of byte elements of that column before changing. The process of mix columns is illustrated in Figure 5.

```
Multiplication Matrix

2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2

16 byte State

b1  b5 b9  b13
b2  b6 b10 b14
b3  b7 b11 b15
b4  b8 b12 b16
```

Figure 5. Multiplication coefficient matrix and input state for mix columns

Each element in the input state is multiplied with a constant matrix (coefficient matrix) as shown in Figure 6, over Galois field, given by equation:

$$\begin{aligned} b1 &= (b1\,x\,2)\ XOR\ (b2\,x\,3)\ XOR\ (b3\,x\,1)\ XOR\ (b4\,x\,1) \\ b2 &= (b1\,x\,1)\ XOR\ (b2\,x\,2)\ XOR\ (b3\,x\,3)\ XOR\ (b4\,x\,1) \\ b3 &= (b1\,x\,1)\ XOR\ (b2\,x\,1)\ XOR\ (b3\,x\,2)\ XOR\ (b4\,x\,3) \\ b4 &= (b1\,x\,3)\ XOR\ (b2\,x\,1)\ XOR\ (b3\,x\,1)\ XOR\ (b4\,x\,2) \end{aligned}$$

The inverse mix columns use a different constant matrix for decryption process.

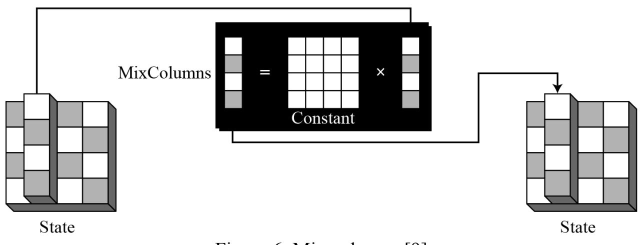

Figure 6. Mix columns [9]

4) *AddRoundKey*

Bitwise XOR of round key to each byte of the array. Round key is derived from the key scheduling operation.

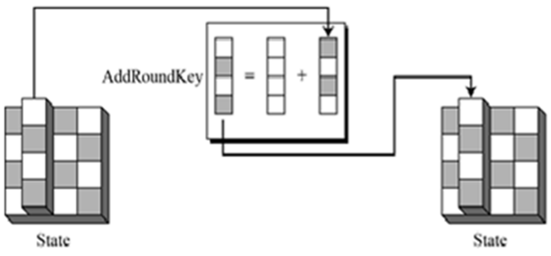

Figure 7. AddRoundKey

Where the round keys are generated and selected through the KeyExpansion of initial key. Key selection in encryption:

$$W_i = SubWord(Rotword(W_{i-1})) \oplus Rcon \oplus W_{i-4} \forall i \text{ multiple of 4}$$
$$W_i = W_{i-1} \oplus W_{i-4} \text{ for Remaining } i$$

III. ARCHITECTURAL DETAILS

There are many design choices encountered during hardware implementation of AES. The choice of technique depends mainly on the target specifications. There is always a trade-off between area and speed when it comes to performance. Greater speed means larger area and vice versa. In this paper we have designed the hardware to improve the throughput. AES encryption consists of 10 rounds which is executed sequentially as the input of the each round is the output of the previous round. To increase the throughput, loop unrolling technique has been implemented. Loop unrolling is the technique of unrolling multiple rounds of operations [10]. Pipelining is the process of placing registers in between so as to increase the throughput at the cost of latency. In this architecture we have implemented pipelining which increased the throughput.

From the initial given cipher key, 9 intermediate keys have been generated by key expansion process which is done in parallel to the encryption process. There are two ways to implement key scheduling. First method is to generate all the required keys in advance or in beginning [10], which will require a huge storage element. The second method is to generate key in parallel, which is more advantageous as it requires less space and has to generate only one key at a time which makes it less computational intensive. We have used the second method where key scheduling operation runs parallel to main encryption operation.

In SubBytes, S-Box (INV S-Box) used is derived from the inverse function over GF (2^8), known to have good non-linearity properties. S-Box (INV S-Box) is implemented using Look Up Table (LUT).

ShiftRows involves the cyclic shift operation has been performed by re-routing the vector inputs to the different outputs.

MixColumns architecture implementation is preferable to perform multiplication in Galois Field of Mathematical Computation. To increase the performance, 16 individual blocks have been implemented in parallel. The Multiplication in Galois Field is done using two Look up Tables (LUT's) E and L tables. Addition is performed by XOR operation. There are two types of exceptions which are encountered due to the use of Galois multiplication. These exceptions are handled in this operation by the following method:

1. Any number multiplied by one is equal to its self and does not need to go through the above procedure [11] – A switch to bypass the value is implemented using multiplexer.
2. Any number multiplied by zero equals zero[11]- to handle this exception we have used a comparator for checking zero, and then a switch to bypass the value to the last step in case of zero value. The switch is implemented using multiplexer.

Apart from these exceptions we have handled overflow, by subtracting the value from 255 whenever the value is more than 255 is encountered.

Since all rounds are dependent we did loop unrolling and pipelined each round. Parallelism was implemented in individual blocks in order to obtain high throughput.

The security factor of the AES Encryption / Decryption Standard mainly depends on this part. For better security, AES Algorithm the first round user key is XORed with the original Plain / Cipher Text. And next round onwards Expanded Key from Expanded Key Schedule is XORed with data. Key Expansion is implemented using LUT and XOR operations.

IV. FPGA IMPLEMENTATION DETAILS

The chosen platform for implementation is a Xilinx Virtex-7 and Virtex-Ultrascale FPGA. Although our paper focus mainly on FPGA, we have synthesized the design in ASIC also.

After designing the complete architecture for encryption and decryption using Synphony Model Compiler on MATLAB-Simulink and the result is verified using standard test vectors from the python implementation of the algorithm.

The design was synthesized using SHLS tool of Symphony Model Compiler for the FPGA boards. Synplify Pro tool was used to obtain the timing report from the synthesized Verilog files.

V. RESULTS

The verification is done using the test vectors as given in Table 3. Initial architecture is done with only loop unrolling and the results for one and two AES blocks are summarized in Table 4 and Table 5 respectively. To achieve the desired throughput pipelining stages have been added to the initial architecture by which the highest throughput value of 120.63 Gbits/sec for two AES blocks on Vertex-Ultrascale was obtained. The results for pipelined design have been briefed in Table 6 and Table 7 respectively. Comparative results obtained by analysis of values of one and two AES blocks are given in Figure 8 and Figure 9.

TABLE 3
EXAMPLE TEST VECTORS

Plain Text	0x3243f6a8885a308d313198a2e0370734
Cipher Key	0x2b7e151628aed2a6abf7158809cf4f3c
Cipher Text	0x3925841d02dc09fbdc118597196a0b32

TABLE 4
ONE AES BLOCK WITHOUT PIPELINING

Device	Frequency (MHz)	Throughput (Gbits/sec)
ASIC	333	39.7
Virtex 7	135	16
Virtex - Ultrascale	219	26.1

TABLE 5
TWO AES BLOCKS IN PARALLEL WITHOUT PIPELINING

Device	Frequency (MHz)	Throughput (Gbits/sec)
ASIC	333	79.4
Virtex 7	135	32.2
Virtex - Ultrascale	217	51.7

TABLE 6
ONE AES BLOCKS IN PARALLEL WITH PIPELINING

Device	Frequency (MHz)	Throughput (Gbits/sec)
ASIC	800	95.3
Virtex 7	235	28
Virtex - Ultrascale	553	65.9

TABLE 7
TWO AES BLOCKS IN PARALLEL WITH PIPELINING

Device	Frequency (MHz)	Throughput (Gbits/sec)
ASIC	800	190.73
Virtex 7	363	86.5
Virtex - Ultrascale	506	120.63

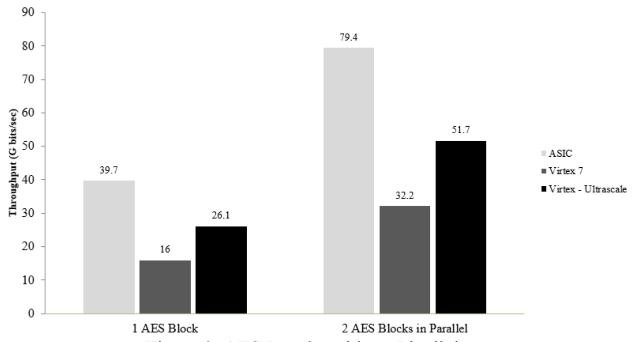

Figure 8. AES Results without Pipelining

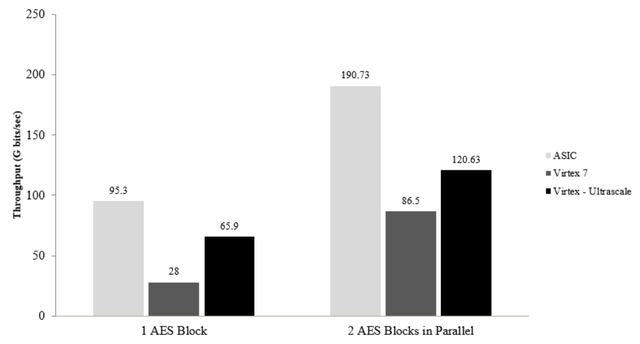

Figure 9. AES Results with Pipelining

VI. CONCLUSION

Improvement in throughput is an assured outcome of hardware implementation of AES algorithm. However, in the case of hardware, there is always a trade-off between area and high speed. A large key size is desired for both security and efficiency. So the work done can be extended to 192 and 256 key sizes.

REFERENCES

[1] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard .

[2] Deshpande, Ashwini M., Mangesh S. Deshpande, and Devendra N. Kayatanavar. "FPGA implementation of AES encryption and decryption."*Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009. 2009 International Conference on*. IEEE, 2009.

[3] Tonde, Ashwini R., and Akshay P. Dhande. "Review paper on FPGA based Implementation of Advanced Encryption Standard (AES) algorithm."*International Journal of Advanced Research in Computer and Communication Engineering* 3.1 (2014).

[4] Manjesh.K.N, R K Karunavathi, "Secured High throughput implementation of AES Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 5, May 2013 ISSN: 2277 128X.
Available: http://www.ijarcsse.com/docs/papers/Volume_3/5_May2013/V3I5-0471.pdf

[5] Progetto AA 08-09: Advanced Encryption Standard. Available:
http://vhdl.pbworks.com/w/page/4302806/Progetto%20AA%2008-09%3A%20Advanced%20Encryption%20Standard

[6] Lan, Luan. "The AES encryption and decryption realization based on FPGA."*Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*. IEEE, 2011.

[7] Lecture Notes on "Computer and Network Security" by Avi Kak (kak@purdue.edu)
Available:
https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf

[8] http://aescryptography.blogspot.com/2012/04/shiftrows-step.html

[9] https://en.wikipedia.org/wiki/Rijndael_mix_columns

[10] Park, Song J. "Analysis of aes hardware implementations." *Department of Electrical and Computer Engineering, Oregon State University* (2003).

[11] Advanced Encryption Standard by Example- Written By: Adam Berent
Available: http://www.adamberent.com/documents/AESbyExample.pdf