High Speed, Pipelined Implementation of Advanced Encryption Standard (AES) on FPGA

Fatemeh Aghazadeh Dizaji[1] and [2] Miftahur Rahman
North South University (NSU), Dhaka, Bangladesh
Department of Electrical Engineering and Computer Science (EECS)
e-mail: f19781211@gmail.com

***Abstract**: Advanced Encryption Standard (AES), the latest publicly announced and strongest ever cryptographic algorithm contains ample parallelism in its structure and is very fast when implemented in dedicated hardware. This exactly is the reason, to choose Field Programmable Gate Array (FPGA) and not the microcontroller as implementation platform. This implementation can be carried out through several trade-off between area and speed. This paper presents an FPGA implementation of 128- bit block and 128-bit key AES cipher. The processor design is completely described in Verilog language. The cipher operates at 279 MHz and consumes 90 clock cycles for encryption and decryption, resulting in a throughput of 402 Mbps. Synthesis result in the use of 944 logic cells .The desire was to achieve the highest possible performance with implementation on Altera's Startix FPGA.*

***Keywords**: Advanced Encryption Standard (AES), Field Programmable Gate Array (FPGA), Cryptography.*

1. INTRODUCTION

Cryptography is of importance in digital communication systems. The security aspects of many applications such as Automated Teller Machines (ATMs), e-commerce, internet bank service depend on various cryptographic schemes. Today, however, the term refers to the science and art of transforming messages to make their transmission secured and immuned to eavesdropping. Due to the increasing usage of internet application and wireless communication, the data security becomes more and more important.

The three major design targets with respect to the hardware realization are: optimization for area or cost, low latency that minimizes time to encrypt a single block and high throughput to encrypt multiple blocks in parallel. All these design criteria involve a trade-off between area and speed. There are a wide range of equipment encryption is needed for authentication and security but throughput is not a principal concern. A low cost, small area design could be used in smart cards application as well as in other storage devices and low speed communication channels.

This paper presents an architecture for 128-bit AES. Algorithm implemented on an Altera FPGA device. The goal of this design is to achieve the highest possible performance with implementation on Altera's Startix FPGA. This paper is organized as follows. Section 2 presents an overview of AES algorithm. Section 3 discusses implementation of AES on FPGA. In section 4, the hardware consumption and test result is provided. Section 5 concludes the paper.

2. AES ALGORITHM

The use of encryption/decryption is as old as the art of communication. The Advanced Encryption Standard (AES) is an encryption algorithm for securing sensitive but unclassified material by U.S. Government agencies. In October 2000, the Rijndael algorithm [1], developed by Joan Daemon and Vincent Rijmen, was selected as the winner of the AES development race. As a replacement of DES, AES is presently widely used in both software and hardware implementations. Hardware approaches are attractive because it provides better throughput as well as higher physical security. Besides, the byte-wise arithmetic, AES gives hardware approaches more convenience.

In [2], the AES algorithm is clearly defined with key, functional blocks, and round numbers. The fixed length of plaintext is 128 bits, the lengths of keys are 128, 192 and 256 bits, and the execution round numbers have 10, 12 and 14. For the example of 128-bit key, during operations, the key must be segmented into 16 bytes, and the segmented 16 bytes will be mapped into a 4×4 matrix, which is called the state matrix. Each byte in the state matrix must be normalized under Galois Field (GF) (2^8) with the modulus of $x^8 + x^4 + x^3 + x + 1$. The AES operations include four transformation calculations, which are SubBytes, ShiftRows, MixColumns and AddRoundKey in order.

2.1 SubBytes Transformation

This is a non-linear transformation that operates independently on each byte of the state using a substitution table (called S-box) [3]. The S-box is a one-to-one mapping table and consequently it is invertible. In the SubBytes step, each byte in the array is updated using an 8-bit S-box. This operation provides the non-linearity in the cipher. This transformation is constructed based on the two following phases:

1. Take the multiplicative inverse in the finite field GF (2^8), (Galois fields) [4],

2. Apply an affine (over GF (2)) transformation defined by

$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i$$

To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a rearrangement), and also any opposite fixed points.

2.2 ShiftRows Transformation

The ShiftRows transformation performs the fixed cyclic byte shift according to different row positions. In the 0th row, this row does not act byte shift. In the first row, this row acts one byte shift. In the second row, this row acts two bytes shift. In the third row, this row acts three bytes shift. The ShiftRows performs cyclic left shift during AES encryptions, and the ShiftRows performs cyclic right shift during AES decryptions.

2.3 MixColumns Transformation

The MixColumns transformation acts the row-by-row mapping operations. During encryptions, the row-by-row operation in based on the mapping polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ under constrains of GF (2^8) and x^4+1 modulus. During decryptions, the row-by-row operation is based on the mapping polynomial $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{0e\}x + \{09\}$ under constrains of GF(2^8) and x^4+1 modulus.

2.4 AddRoundKey Transformation

The AddRoundKey transformation performs the bit-by-bit XOR operations between outputs of MixColumns and the round key.

3. IMPLEMENTATION AND DISCUSSION

ASE proposed in this paper is aimed to realize the high speed. To make AES suitable to high-speed data conditions, we need to optimize the architecture. Meanwhile by sharing resource and eliminating common sub expression we can reduce the use of the hardware resource.

There are three basic architectures of AES to improve the throughput: Loop unrolled, pipelined, sub-pipelined. The Loop unrolled architecture, it is the architecture that don’t buffer the data but input the data to the next round function directly. The pipeline architecture which buffers the data among round functions. Finally the sub pipelined architecture that buffers the data among round functions and among inner transformations. That’s why we adopt the pipelined architecture.

Designing a high speed low area S-Box and inverse S-Box is one of the most critical problems in the research process, because the realization of S-box/InvS-box is the only nonlinear transformation in the four transforms of AES arithmetic and is the key point to improve the throughput of AES cipher core and decrease the resource used to implement the AES. Most of earlier designs implemented the byte transformation using look up table techniques, which put all the transformed data in BRAMs and output the transformed data recording to the data you want to implement byte transformation. But the design of pipelined AES needs many S-boxes/InvS-boxes, requiring 200 S-boxes/InvS-boxes in ten rounds transformation and key expansion totally. Using look up table techniques not only use more hardware resource (in terms of memory) but also limit maximum operable clock frequency to BRAMS included in Field Programmable Gate Array (FPGA). So in this paper, we design the S-box/InvS-box in composite field of GF (2^8).

In our design, both encryption and decryption are included, and decryption can be realized in inverse transformations of encryption process. But if we use simple inverse encryption process to decrypt, we need to reorganize the implementation modules in pipelined architecture, and it will spend much more hardware resource, so the equivalent decryption is used according to the equivalent encryption. InvMixColumns transformation must be used to process the expanded key coming from key expansion process in decryption. But the InvMixColumns transformation is programmed separately, not using the MixColumns module in round transformation. In this way, we can get higher transformation speed but use less hardware resource. The flow of key expansion is shown in Fig.1.

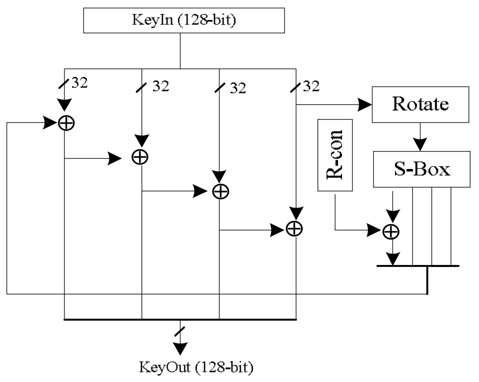

Fig.1. KeyExpansion

3.1 Integration of Mix column and InvMix column function.

The function of mix/Inv-mix column can be described by the following equation:

Encryption output = [2, 3, 1, 1] * $(b_0, b_1, b_2, b_3)^T$; (1)

Decryption output = [E, B, D, 9] * $(b_0, b_1, b_2, b_3)^T$; (2)

Equation (2) can be expressed in the following matrix form first mentioned by Satoh [5] [6].

$$\begin{bmatrix} i_0 \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} e & b & d & 9 \\ 9 & e & b & d \\ d & 9 & e & b \\ b & d & 9 & e \end{bmatrix}$$

$$\otimes \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \left(\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 \end{bmatrix} + \begin{bmatrix} 4 & 0 & 4 & 0 \\ 0 & 4 & 0 & 4 \\ 4 & 0 & 4 & 0 \\ 0 & 4 & 0 & 4 \end{bmatrix} \right) \otimes$$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix} + \begin{bmatrix} 8b_0 & 8b_1 & 8b_2 & 8b_3 \\ 8b_0 & 8b_1 & 8b_2 & 8b_3 \\ 8b_0 & 8b_1 & 8b_2 & 8b_3 \\ 8b_0 & 8b_1 & 8b_2 & 8b_3 \end{bmatrix} + \begin{bmatrix} (4b_0 + 4b_2) \\ (4b_1 + 4b_1) \\ (4b_0 + 4b_2) \\ (4b_1 + 4b_2) \end{bmatrix}$$

In our design, we express the operation of InvMixColumn as:

output [127:120]= 2(2 (2(in[127:120] ⊕ in [119:112]) ⊕ 2(in[111:104] ⊕ in[103:96] ⊕ in [127:120]))) ⊕ 2(in [127:120] ⊕ in [119:112]) ⊕ in [111:104] ⊕ in[103:96] ⊕ in [119:112]).

So, an efficient design of MixColumn and InvMixColumn transformation are shown in Fig.2 and 3 respectively.

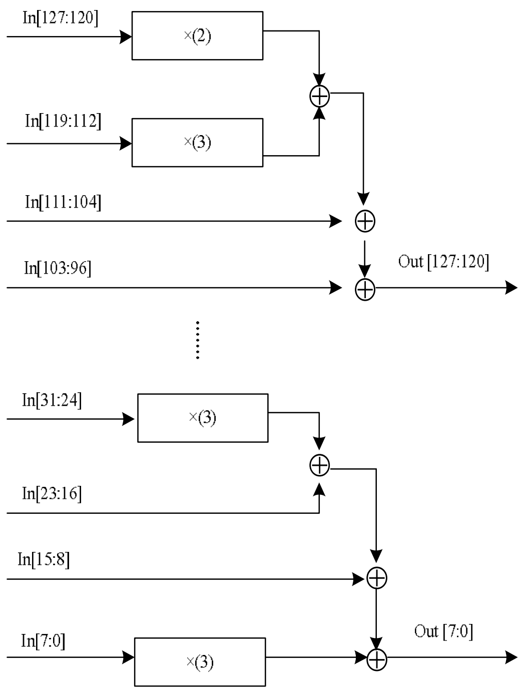

Fig.2. Design of MixColumn

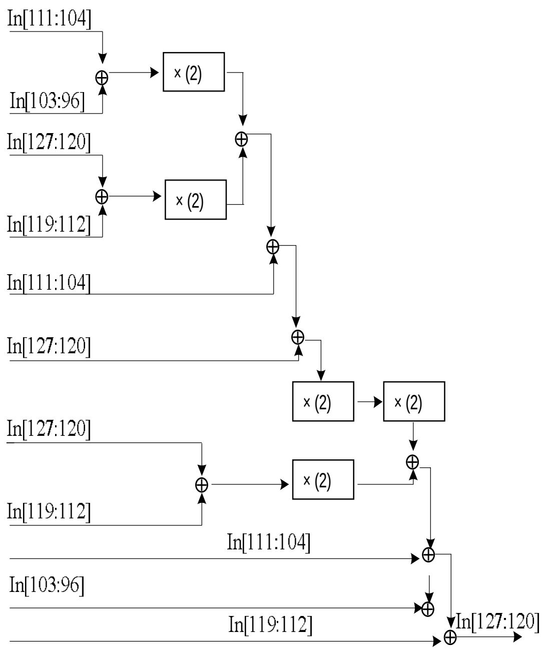

Fig.3. Design of InvMixColumn

4. PERFORMANCE AND COMPARISION

The AES architecture described above has been implemented using Verilog HDL. We applied pipelining technology in both encryptor and decryptor keyschedule modules to optimize the speed/area ratio, which achieves 0.42 Mbps/Slice in Startix EP1S20F780C5. From the result, the design performs is 328.46 Mb/s for encryption and 475.2 Mb/s in decryption. The clock frequency used is 320.95MHz with clock period of 4.33 ns for encryption, also for decryption are 326.80 MHz and 3.06 ns respectively.

Our proposed AES architecture provides the throughput of 402 Mbps and clock frequency of 279 MHz. Compared with similar previous works, our proposed low-cost and efficient AES architecture only uses 944 slices, and achieves the throughput of 402 Mbps when implemented in Startix EP1S20F780C5. The throughput/area ratio is 0.42, which is relatively high in low- cost designs.

The design in this research is also targeted for low speed and space restriction application. With the space requirement restriction, it would be hard to achieve more than 1Gb/s but it would possible if pipeline strategy is used. Although our design comparison is slightly slow, but we need to look into deeper considerations on other implementation listed in Table 1 most designs reported use a big size of FPGA and they used almost all gates and resources on the FPGA. The proposed design can be efficiently applied in computing resources restricted environments, such as wireless devices and embedded devices.

Table1: COMPARISON OF AES DESIGN ON FPGA

Design	Slice (Area)	Device	Throughput	Frequency
Hodjat et al. [7]	9446	VIRTEX2P XV2VP20	21.54 Gb/s	169.1 MHz
Lemsitzer et al. [8]	7300	VIRTEX4 FX100	3500 Mb/s	110 MHz
Bules et al. [9]	1800	SPARTAN3	1700 Mb/s	150 MHz
Proposed design	944	STARTIX EP1S20F780C5	402 Mb/s	279 MHz

6. CONCLUSIONS

This paper presents a compact reconfigurable FPGA architecture for the AES implementation. We applied pipelining technology in both encryptor and decryptor keyschedule modules to optimize the speed/area ratio, which achieves 0.4 Mbps/Slice in Startix EP1S20F780C5. Also we changed the structure of the key expansion, MixColumn and design a different control unit. The R-con part produces numbers proportional according to the same level that increases the speed of control unit. The throughput of our proposed design achieves 402 Mbps and maximum frequency of 279 MHz.

REFERENCES

[1] J. Daemen and V. Rijmen, “AES Proposal: Rijndael”, version 2, 1999. Available on: http://www.esat.kuleuven.ac.be/~rijmen/rijndael.

[2] “Advanced Encryption Standard (AES)” Federal Information Processing Standards Publication 197, Nov. 26, 2001.

[3] Rijmen Vincent, "Efficient Implementation of the Rijndael S-box," Available on: http://www.iaik.tugraz.ac.at/research/krypto/AES/old/-ri jmen/rijndael/sbox.pdf.

[4] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.

[5] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat,"Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications",Information Technology Coding and Computing, 2004.Proceedings. ITCC 2004, Volume 2, pp.583 – 587, Vol.2, 2004.

[6] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware ArchitectureWith S-Box Optimization," in Proc. LNC ASIACRYPT'01, vol. 2248, pp. 239-254, Dec. 2001.

[7] A.Hodjat and I. Verbauwhede. A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA. In FCCM ’04: Proceedings of the 12th Annual IEEE Symposium on Field Programmable Custom Computing Machines, pp.308–309, Washington, DC, USA, 2004. IEEE Computer Society.

[8] S. Lemsitzer, J. Wolkerstorfer, N. Felber, and M. Braendli. Multigigabit GCM-AES Architecture Optimized for FPGAs. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pp. 227– 238. Springer, 2007.

[9] P. Bulens, F. Standaert, J. Quisquater, P. Pellegrin, and G. Rouvroy. Implementation of the AES-128 on Virtex-5 FPGAs. In *Progress in Cryptology - AfricaCrypt 2008*, pp.16 – 26. Springer, 2008.