ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE)
Vol 3, Issue 5, May 2016

FPGA Implementation of High Throughput Dual Key based AES Encryption and Decryption

[1] Naveen Kumar M S [2] Manjunath C Lakhannavar
[1][2]Department of E & C,
M. S. Ramaiah Institute of Technology

Abstract: **---- Data security plays a major role in today's technology. Cryptography is one of the industry standards in providing data security since many years. Federal Information Processing Standard (FIPS) is approved Advance Encryption Standard (AES) cryptographic algorithm that can be used to protect electronic data. But the conventional scheme of AES is vulnerable for cryptanalysis, due to static S-box which will never vary with the input text/key. Noticeable drawbacks with AES are, it can only support one key, and is prone to easy reverse engineering which can lead to insecure data. Thus S-box value is necessary for changes in input key. In this paper, a new scheme of AES is discussed which involves in the generation of Key based S-Boxes, with dual key AES and implemented in pipeline architecture to improve the throughput with low latency.**

***Keywords*—AES, Encryption, Decryption, S-Box, Dual key, Pipeline, Cryptography, Security**

I. INTRODUCTION

FIPS accepted Advanced Encryption Standard in November 2001 so as to provide digital data security. According to size of the encryption key different versions of AES (AES128, AES196, and AES256) has been developed. Depending on the versions, the number of rounds executed in the process varies accordingly. In this paper, a hardware model for implementing the AES 128 algorithm was developed using the Verilog hardware description language. A unique feature of the proposed design is dynamic S-Box with Dual AES implemented in pipeline architecture. This design is more secure and has high throughput with low latency when compared with the existing design.

II. CONVENTIONAL AES ALGORITHM

The AES is a block cipher, which operates on 128 bits key and 128 bits data block. The input to each round consists of a block of message called the state and the round key as shown in figure 1. The round key changes in every round which is generated by key expansion algorithm.

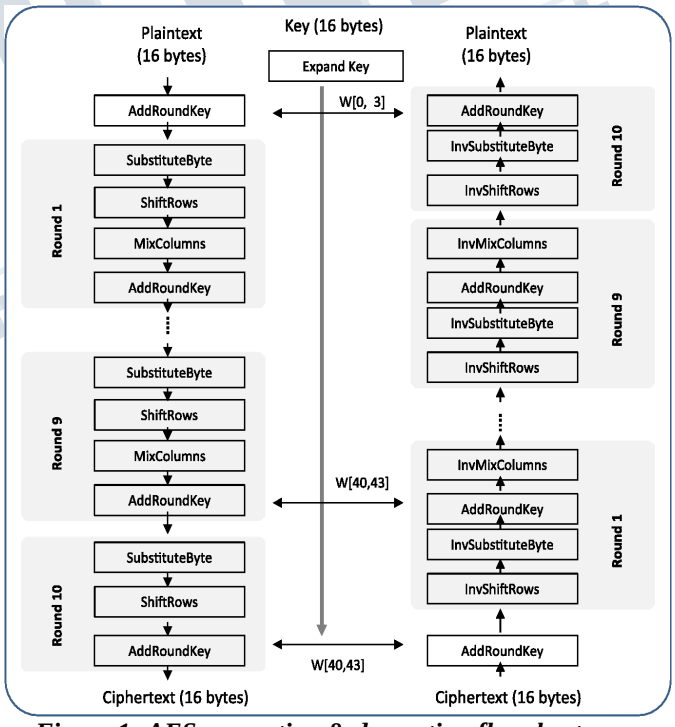

Figure 1: AES encryption & decryption flowchart.

The state can be represented as a rectangular array of bytes. This array has four rows and four columns (1). The same could be applied to the cipher key. The cipher consists of a 10 rounds. Each round of AES encryption function consists mainly of four different

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE)
Vol 3, Issue 5, May 2016

transformations: subbyte, shiftrow, mixcolumn and key addition. On the other hand, each round of AES decryption function consists mainly of four different transformations: invsubbyte, invshiftrow, invmixcolumn, and key addition**.**

$$\text{State} = \begin{bmatrix} d_{15} & d_{11} & d_7 & d_3 \\ d_{14} & d_{10} & d_6 & d_2 \\ d_{13} & d_9 & d_5 & d_1 \\ d_{12} & d_8 & d_4 & d_0 \end{bmatrix} \quad \text{..... (1)}$$

The description of the four transformations of the cipher and their inverses will be given below.

A. ***Sub byte transformation*****:** sub byte transformation is a non-linear byte substitution transformation, where each element of the state matrix is replaced by a new element from a pre calculated s-box table. That s-box table contains 256 numbers (from 0 to 255) and their corresponding resulting values. The main complexity of the algorithm lies in this transformation. The implementation of this transformation is very simple and hence all the modifications presented in this paper are on this transformation.

B. ***Inverse sub byte transformation***: the invsubbyte transformation is done using a once pre-calculated substitution table called invsbox. That table (or invs-box) contains 256 numbers (from 0 to 255) and their corresponding values.

C. ***Shift Row Transformation*****:** The rows of the state matrix are cyclically left shift but each row is shifted with a different offset. Row *i* is shift over (*i*-1) byte offset.

D. ***Inverse Shift Row Transformation***: The rows of the state are cyclically right shift over different offsets. Row *i* is shift over (*i*-1) byte offset.

E. ***Mix Column Transformation****:* The state matrix is multiplied with a constant matrix (2) to obtain the new matrix. Matrix multiplication is done over Galois Field. In this transformation, the bytes are treated as a polynomials rather than numbers. The implementation complexity of this transformation is high and hence this algorithm is also used as it is.

$$\begin{pmatrix} S_1^* \\ S_2^* \\ S_3^* \\ S_4^* \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \bullet \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{pmatrix} \quad \text{..... (2)}$$

F. ***Inverse Mix column Transformation****:* This is similar to mix column transformation, the state matrix is multiplied with a constant matrix (3) to obtain the new matrix.

$$\begin{pmatrix} S_1^* \\ S_2^* \\ S_3^* \\ S_4^* \end{pmatrix} = \begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix} \bullet \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{pmatrix} \quad \text{.....(3)}$$

G. ***Add Round Key****:* This transformation involves a bitwise XOR operation between the state array and the result of Round Key that is output of the Key Expansion algorithm

III. PROPOSDE WORK

A. Generation of Dynamic Key Based S-Box

The architecture for generation of s-box and inverse s-box dynamically is proposed in [2]. The subbyte transformation is computed by taking the multiplicative inverse in GF (2^8) followed by an affine transformation. For its reverse, the invsubbyte transformation, the inverse affine transformation is applied first prior to computing the multiplicative inverse. Sub-byte: multiplicative inversion in GF (2^8) → affine transformation (4)

Inverse sub-byte: inverse affine transformation → multiplicative inversion in GF (2^8) (5)

The affine transformation (4) and its inverse can be represented in matrix form and it is shown below.

$$AT(a) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad (6)$$

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE)
Vol 3, Issue 5, May 2016

$$AT^{-1}(a) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad (7)$$

The at and at^{-1} are the affine transformation and its inverse while the vector *a* is the multiplicative inverse of the input byte from the state array.

The multiplicative inverse is generate in GF (2^8) using irreducible polynomial

$$X^8+X^4+X^3+X^1+1 \quad \ldots\ldots (8)$$

Represented as 100011011 in the form of bits

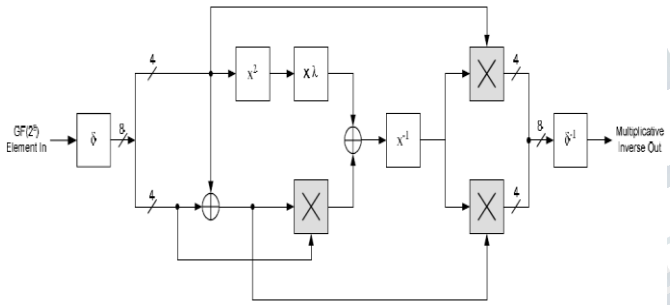

Figure 2(a) multiplicative inversion for the s-box in GF (2^8).

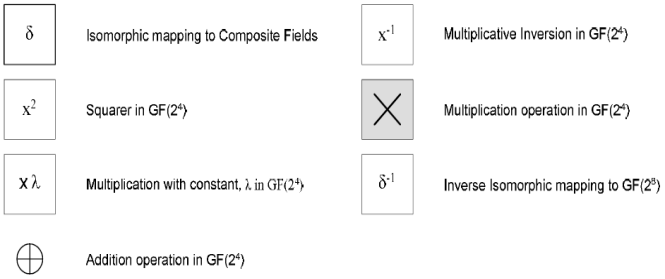

Figure 2(B) The Building Block Of Multiplicative Inverse

The multiplecative inverse calculation is carried out by mapping the GF (2^8) to GF (2^4) as shown in figure 2 (a), (b).

To generate a key based s-box, the system takes an another 8-bits input key called skey, this skey is used as a address to the lookup table which contains set of 128 bits system keys. This system key is used to generate a 8-bit offset. The flow chart of algorithm used to generate offset is show in figure 3.

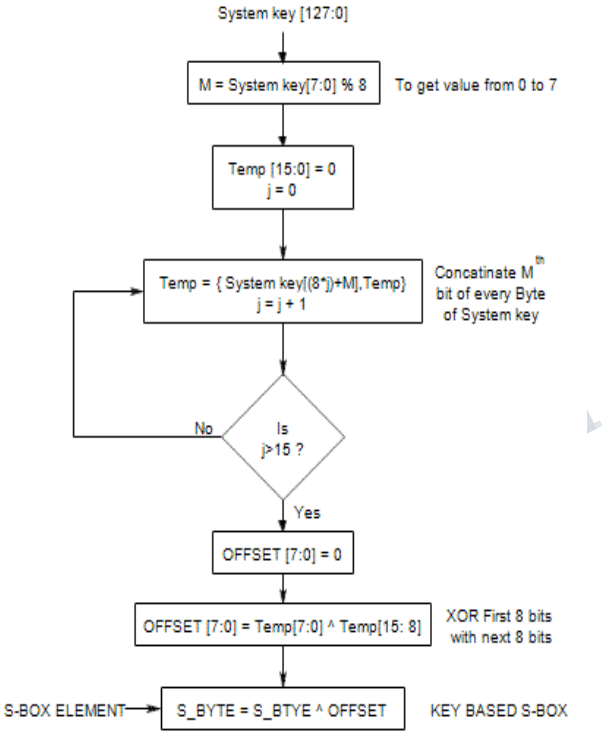

Figure 3 flowcharts for generation of key s-box

In the above algorithms, the 8 bit input SKEY selects 128 bit system key from the lookup table, this system key split into 16 sets of 8 bits, the first 8 bits is modulo by 8 and the remainder is stored in m variable. Now m^{th} bit of each byte of system key is stored in 16 bit temp variable. Offset is calculated by ex-or the first 8 bits of temp variable with the next 8 bits of temp variable. Finally the key based s-box is obtain by ex-or s-box element with offset. Similarly the key based inverse s-box is obtained by EXOR inverse s-box element with offset.

Thus proposed key based s-box is more complex and secure than the existing static s-box.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE)
Vol 3, Issue 5, May 2016

B. Pipeline method of implementing AES the pipeline method of implementing the 11 rounds of AES Encryption Is Shown In Figure 4

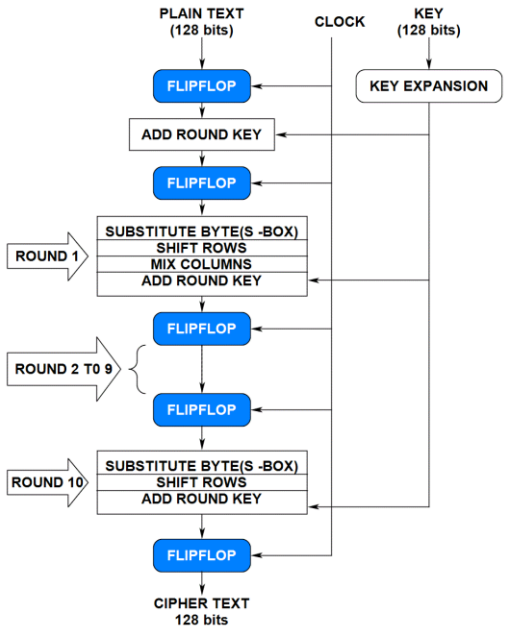

Figure 4 Pipeline Method Implementation Of AES Encryption

In pipeline method, between each round of encryption process flip-flops are used to store and passing encryption data to next round on active clock edge. In each round, proper scheduling to data and key form key-expansion block to avoid timing violations. Using key based s-box, pipeline AES encryption is implemented on virtex 7 xc7v585t, operates at 212 MHz.

To implement AES decryption in pipeline method is difficult because the AES decryption process starts with cipher text and key which is used in the last round of encryption.

So key expansion and AES decryption core is implemented separately, first 11 cycles are used generate complete 10 keys and stored in memory which required for decryption, then decryption process starts which next takes 11 clock cycle to decrypt the cipher text using key based inverse s-box, pipeline AES decryption is implemented on virtex 7 xc7v585t, operates at 192 mhz.

The pipeline method of implementing AES decryption is shown in figure 5

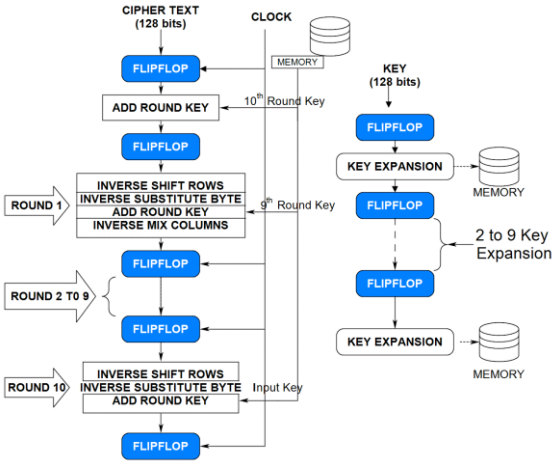

Figure 5 pipeline method implementation of AES decryption

IV. SIMULATION AND SYNTHESIS RESULTS

The architecture proposed above was compiled and synthesized on virtex 7 xc7v585t fpga using ise navigator and simulated using isim tool, the results were found to be as follows:

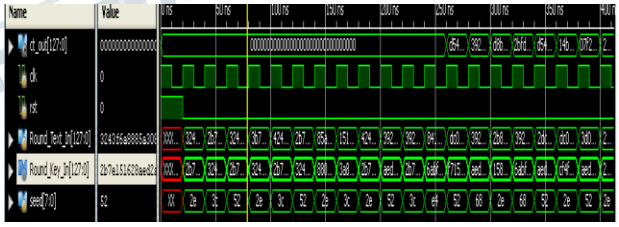

figure 6 simulation of dual key encryption

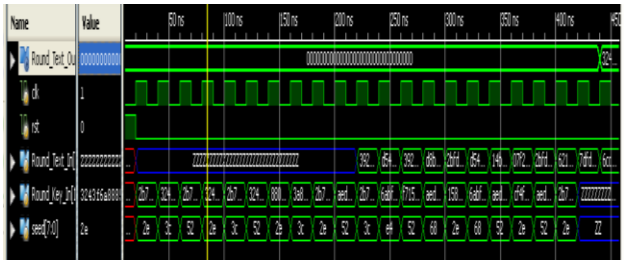

Figure 7 Simulation of dual key Decryption

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE)
Vol 3, Issue 5, May 2016

Table1 Synthesized result

Device: xc7v585t	Proposed Encryption	Proposed Decryption
No. of Slice Register	3,034	5,658
No. of Slice LUTs	17,249	22,542
No. of fully used LUT-FF pairs	2,641	4,965
No. of bonded IOBs	394	394
No. of BUFG/BUFGCTRLs	2	2
No. of Memory	0	1,152
Maximum operating Freq.	212 MHz	192MHz
Latency	11 cycles	22 cycles
Throughput	27.16 Gbps	24.57 Gbps

V CONCLUSION

Results shown in table1 indicates that the encryption algorithm is operating with a maximum of 212mhz clock cycles, with latency of 11 clock cycles and throughput of 27.16gbps. Decryption algorithm is operating with a maximum of 192mhz clock frequency with a latency of 22 clock cycles and throughput of 27.16 gbps. With this proposed methodology, high throughput and secured data encryption can be attained.

REFERENCE

1) Advanced Encryption Standard, Federal Information Processing Standards 197", National Institute of Standards and Technology, November 2001

2) A.F Webster and S.E Travares, "On The Design of S-boxes," Queen's university Kingston, Springer- verlag, Canada1998

3) Muhammad Asim, "Efficient and Simple Method for Designing Chaotic S-boxes" Electronic and Telecommunications Research Journal, University of Technology Petronas, Malaysia February 2008.

4) Xinmiao Zhang and Keshab K. Parhi, "On the Optimum Constructions of Composite Field for the AES Algorithm", IEEE, VOL. 53, NO. 10, OCTOBER 2006

5) F. Fahmy and G. Salama, "A proposal for Key-dependant AES, " 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT), TUNISIA March 2005.

6) Joan Daemen and Vincent Rijmen."Two-Round AES Differential". Cryptology ePrint Archive, Report 2006/039, 2006.