International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 12, December 2016, PP 43-51
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Design Implementation of Composite Field S-Box using AES 256 Algorithm

Challa Vamshi Krishna [1], N. Shivakumar [2], Dr. D Subba Rao [3]

[1]Department of ECE, Siddhartha Institute of Engineering and Technology, Hyderabad, India (PG Scholar)
[2]Department of ECE, Siddhartha Institute of Engineering and Technology, Hyderabad, India (Assistant Professor)
[3]Department of ECE, Siddhartha Institute of Engineering and Technology, Hyderabad, India (Head of the Department for Electronics and Communication Engineering)

ABSTRACT

An efficient implementation of the Advanced Encryption Standard (AES) Algorithm. The presented architecture is adapted for AES encryption, encryption/decryptiondesigns.TheSub, InvSubBytes operations are implemented using composite field arithmetic. Efficient architecture for performing the mix columns & inverse mix columns operation, which is the major operation in the Advanced Encryption Standard (AES) method of cryptography. In the implementation of this AES-256 algorithm has a plaintext of 128 bits and key of 256 bits size. The number of rounds of operations in AES- 256 is 14. The key generation process of AES 256 is different from other AES algorithms. Xilinx 13.2i software is used for simulation and optimization of the synthesizable VERILOG code. All the transformations of both Encryption, Encryption/Decryption designs

Keywords: Encryption/Decryption

INTRODUCTION

The Advanced Encryption Standard, in the following referenced as AES, is the winner of the contest, held in 1997 by the US Government, after the Data Encryption Standard was found too weak because of its small key size and the technological advancements in processor power. Fifteen candidates were accepted in 1998 and based on public comments the pool was reduced to five finalists in 1999[1]. In October 2000, one of these five algorithms was selected as the forthcoming standard: a slightly modified version of the Rijndael. The Rijndael, whose name is based on the names of its two Belgian inventors, Joan Diemen and Vincent Rijmen, is a Block cipher, which means that it works on fixed-length group of bits, which are called blocks. It takes an input block of a certain size, usually 128, and produces a corresponding output block of the same size. The transformation requires a second input, which is the secret key. It is important to know that the secret key can be of any size (depending on the cipher used) and that AES uses three different key sizes: 128, 192 and 256 bits. To encrypt messages longer than the block size, a mode of operation is chosen, which I will explain at the very end of this tutorial, after the implementation of AES. While AES supports only block sizes of 128 bits and key sizes of 128, 192 and 256 bits, the original Rijndael supports key and block sizes in any multiple of 32, with a minimum of 128 and a maximum of 256 bits.

Address for correspondence:

Avamshi.challa409@gmail.com

AES ALGORITHM

The AES algorithm operates on 128-bit data blocks using a cipher key of possible lengths 128/192/256-bits throughout 10/12/14 iterative rounds respectively [2]. Each round consists of a set of transformations namely: Sub Bytes, Shift Rows, Mix Columns, AddRoundKey or their corresponding inverses during decryption.

a. Addround Key

In this operation, a given data input (128 bits) is bitwise XORed with User defined Key(128 bits) to generate a cipher text of 128bits.

b. Subbytes/Inversebytes Transformation using CFA

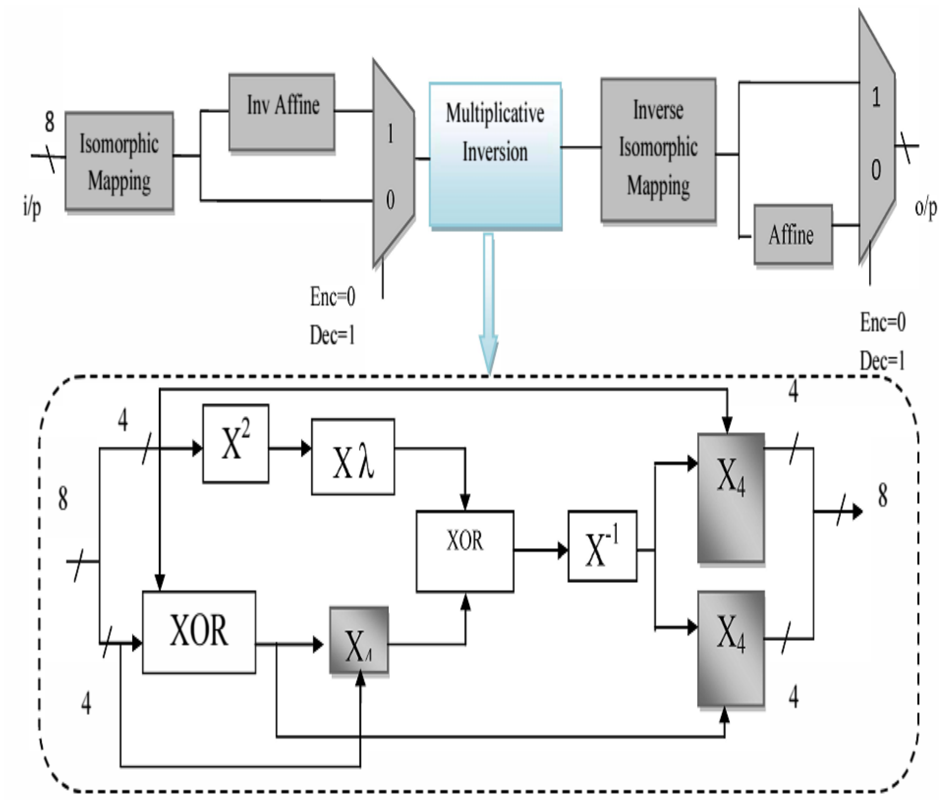


Fig1. *Subbytes and Invsubbyte Block Diagram*

S-box has the main two transfonnations. One is the multiplicative inversion and another one is the affine transfonnation. Fig shows subbyte and inv sub byte In Sub bytes the operation is multiplicative inversion to affine transformation[3]. In inv sub bytes, the operation is inv affine transfonnation to multiplicative inversion. Affme Transformation (AT): The matrix multiplication followed by the addition of a vector is affine transformation. The sum of multiple rotation of byte is a vector. Here the addition operation is the XOR operation. Inv Affine Transformation (ATI): The reverse process is inverse affine transformation. Multiplicative Inversion: Composite field of GF(2^8) cannot directly apply through the multiplicative inversion. The computation process is made by the decomposing the complex form of GF(2^8) in the lower order form of GF(2^2), GF(2^1) and GF((2^2)2). The irreducible polynomial used. to go for several arithmetic operations like squaring, multiplication, inversion and addition. Multiplicative inversion is the costliest field. These are simplified by the simply XOR-AND gates[4].

c. Shift Row/Inverse Shift Row Operation

Shift Row Operation

In this operation, each row of the state is cyclically shifted to the left, depending on the row index.

- The 1st row is shifted 0 positions to the left.
- The 2nd row is shifted 1 position to the left.
- The 3rd row is shifted 2 positions to the left.
- The 4th row is shifted 3 positions to the left.

Inverse Shift Row Operation

In this operation, each row of the state is cyclically shifted to the right, depending on the row index.

- The 1st row is shifted 0 positions to the right.
- The 2nd row is shifted 1 position to the right.
- The 3rd row is shifted 2 positions to the right.
- The 4th row is shifted 3 positions to the right.

d. Mix/Inverse Mix columns Operation

Mix Coloums

Mix columns (MC) and inverse mix columns (IMC) are Implemented by performing matrix multiplication over Galois Field i.e. GF (2^8) using the irreducible polynomial x8 + x4 + X3 + x+ 1. The constant matrices used for mix columns.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} M_1 & M_5 & M_9 & M_{13} \\ M_2 & M_6 & M_{10} & M_{14} \\ M_3 & M_7 & M_{11} & M_{15} \\ M_4 & M_8 & M_{12} & M_{16} \end{bmatrix}$$

Inv Mix Columns

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \bullet \begin{bmatrix} IM_1 & IM_5 & IM_9 & IM_{13} \\ IM_2 & IM_6 & IM_{10} & IM_{14} \\ IM_3 & IM_7 & IM_{11} & IM_{15} \\ IM_4 & IM_8 & IM_{12} & IM_{16} \end{bmatrix}$$

Final round, the Mix Column operation is omitted[5][6].

e. Key Expansion Architecture 128bit

The key expansion algorithm computes each 128-bit round key Kr = (wr,0, wr,1, wr,2, wr,3) column by column using the following equations:

wr,0 = Rot Word(Sub Word(wr-1,3)) + wr-1,0 + RCON[r], (1)

wr,i = wr,i-1 + wr-1,i , for i = 1, 2, 3, (2) where *r* expresses the round number from 1 to 10 with *K0* being the input cipher key, RCON[r] is the hexadecimal value{00,00,00,xr-1} and wr, i is the 32-bit word column *i* in *Kr*. The Sub Word function in (1) performs the Sub Bytes transformation on each of the four bytes of the column *wr-1,3*. The conventional hardware implementation of (1) and (2) is realized by the Sub Word function followed by a successive *XOR*ing to calculate each column wr, i from the previous column wr,i-1. The proposed Key Expansion architecture computes the successive XORing in parallel with the Sub Word function as shown in Fig. 2. Consequently, the remaining part is only to *XOR* the output of Sub Word function with all four columns in parallel. Therefore, the critical path delay is decreased by 3 *XOR* gates through the parallelization of the Key Expansion steps compared to the conventional Key Expansion structure in [7].

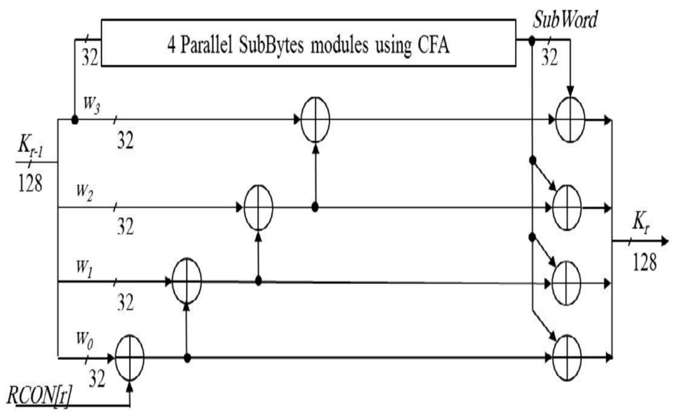


Fig2. *Key Expansion Architecture*

f. Key Expansion Architecture 256bit

Description of the AES-256 Encryption algorithm

1. Key Expansion round keys are derived from the cipher key using Rijndael's key schedule.
2. Initial Round
   - i. Add Round Key each byte of the state is combined with the round key using bitwise xor.
   - ii. Sub Byte a non-linear substitution step where each byte is replaced with another according to a lookup table.
   - iii. Shift Rows a transposition step where each row of the state is shifted cyclically a certain number of steps.
   - iv. Mix Columns is a mixing operation which operates on the columns of the state, combining the four bytes in each column.
   - v. Add Round Key
3. Rounds(2nd to 13th)
   - i. Sub Bytes
   - ii. Shift Rows
   - iii. Mix Columns
   - iv. Add Round Key.

4. Final Round (no Mix Columns)

    i. Sub Bytes

    ii. Shift Rows

    iii. Add Round Key.

In AES 256 the process of generating the key is each round key is a 256-bit array generated as follows,

Input key of 256 bit is divided into eight parts of 32 bits as columns in a matrix4*8. Last column is taken and given as input to S box. The output of S box is given shift rows operation. The above output MSB side 8 bits are xored with the round constant (i.e. round constant value is different for different rounds). The above output is xored with 0th column of input key .The above output is taken as 0th column of the generated new key. 0th column of new key is xored with 1st column of input key gives 1st column of new key. 1st column of new key is xored with 2st column of input key gives 2nd column of new key. 2nd column of new key is xored with 3rd column of input key gives 3rd column of new key. The above obtained 3rd column of new key is given to S box. The output of S box is xored with 4th column of input key which gives 4th column of new key. 4th column of new key is xored with 5th column of input key which gives 5th column of new key. 5th column of new key is xored with 6th column of input key which gives 6th column of new key. 6th column of new key is xored with 7th column of input key which gives 7th column of new key. In this way we generate new keys of 256 bits in AES 256 algorithm by attaching the eight obtained columns of new key.

Encryption Round Architecture

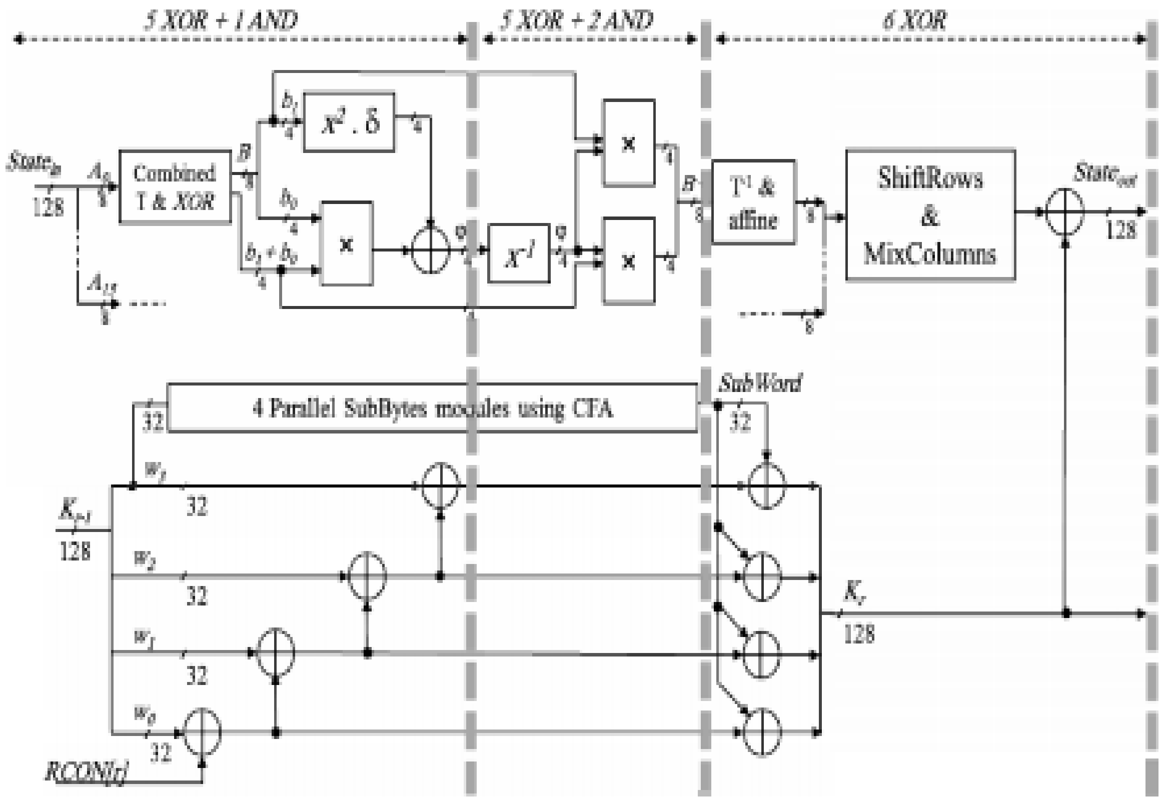


Fig4. *AES Encryption Round Architecture128bitkey*

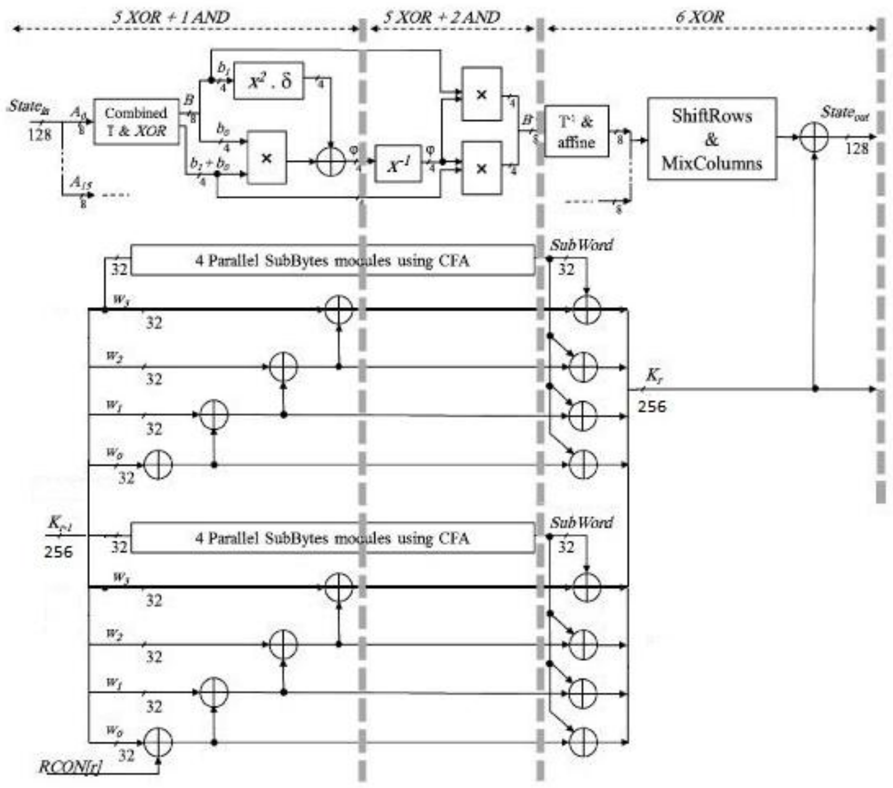


Fig5. *AES Encryption Round Architecture256bitkey*

AES round architecture adopts the sub pipelining technique through the insertion of three-level registers to break down the critical path delay. The placement of the registers is chosen such that the resulting stages delays are balanced while trying to optimize the number of pipelining registers used. Fig. 4 shows the complete sub-pipelined AES encryption round architecture. The vertical grey dashed lines represent the added sub-pipelining registers. The resulting stages delays are provided on the top of each stage respectively as illustrated in Fig. 4[9].

Encryption and Decryption Round Architecture

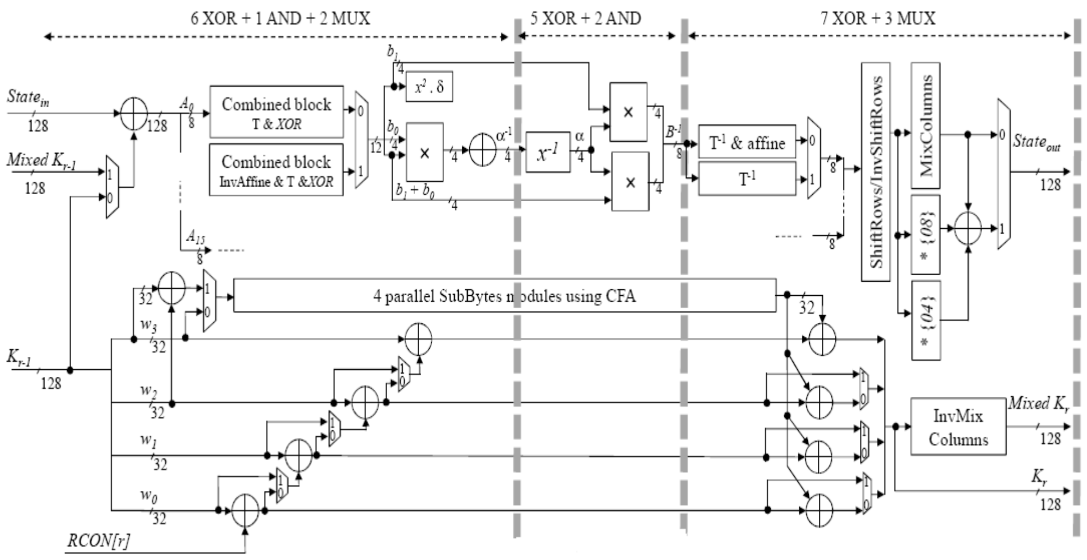


Fig6. *AES Encryption /Decryption Round Architecture128bitkey*

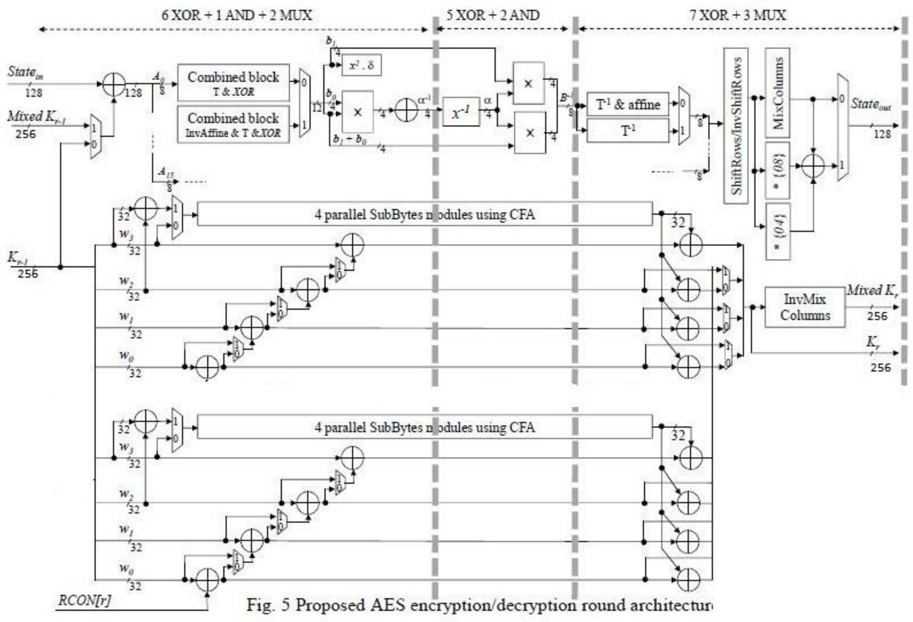
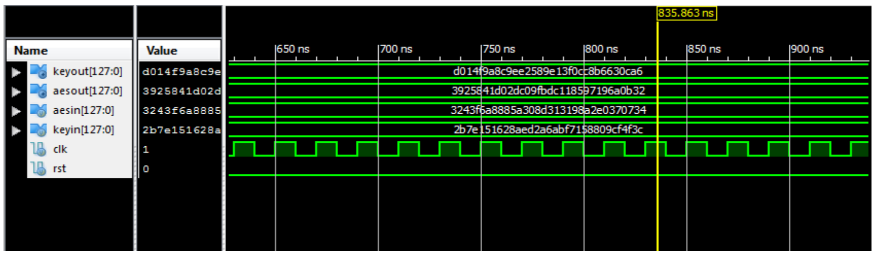


Fig7. *AES Encryption /Decryption Round Architecture256bitkey*

AES decryption process performs the inverse encryption transformations in the following reverse order: Add Round Key, In Mix Columns, Inv Shift Rows, Inv Sub Bytes. Furthermore, the parallel Key Expansion is executed in the reverse order starting from the final key value. In order to follow the same transformations order of the encryption procedure, the *AES* decryption is implemented using the equivalent Inverse cipher method

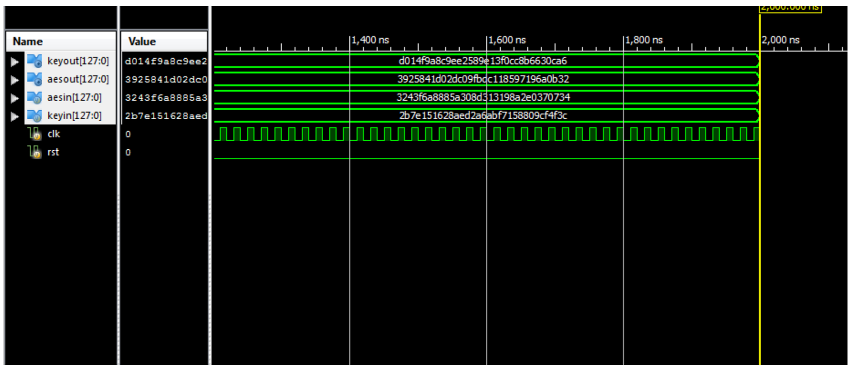
SIMULATION WAVEFORMS

The simulation of AES encryption and encryption/decryption round architectures is done using Xilinx 13.2 ISE design suite. The output wave form is show in below.



Name	Value	
keyout[127:0]	d014f9a8c9e	d014f9a8c9ee2589e13f0cc8b6630ca6
aesout[127:0]	3925841d02d	3925841d02dc09fbdc118597196a0b32
aesin[127:0]	3243f6a8885	3243f6a8885a308d313198a2e0370734
keyin[127:0]	2b7e151628a	2b7e151628aed2a6abf7158809cf4f3c
clk	1	
rst	0	

Fig8. *Simulation of AES Encryption Algorithmwith128bitkey*



Name	Value	
keyout[127:0]	d014f9a8c9ee2	d014f9a8c9ee2589e13f0cc8b6630ca6
aesout[127:0]	3925841d02dc0	3925841d02dc09fbdc118597196a0b32
aesin[127:0]	3243f6a8885a3	3243f6a8885a308d313198a2e0370734
keyin[127:0]	2b7e151628aed	2b7e151628aed2a6abf7158809cf4f3c
clk	0	
rst	0	

Fig9. *Simulation of AES Encryption Algorithmwith256bitkey*

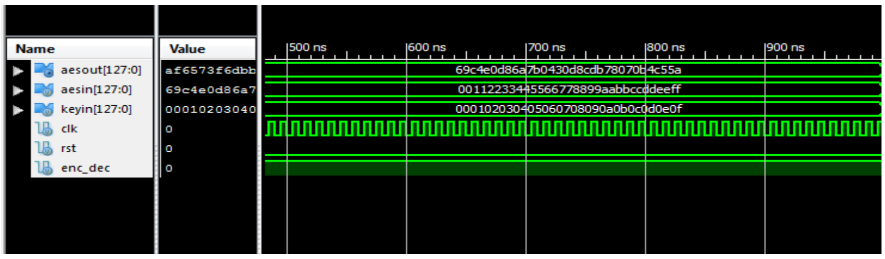


Fig10. *Simulation of AES Encryption/Decryption Algorithm with 128bit*

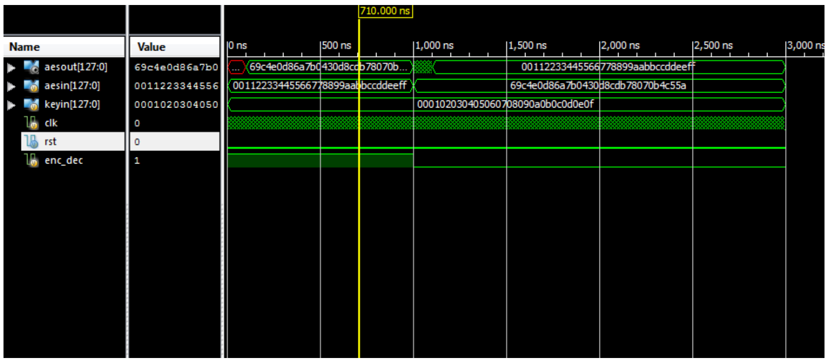


Fig11. *Simulation of AES Encryption/Decryption Algorithm with 256bit*

CONCLUSION

The Advanced Encryption Standard algorithm is an iterative private key symmetric block cipher that can process data blocks of 128 bits through the use of cipher keys with lengths of 128, 192, and 256 bits. An efficient FPGA implementation of 128 bit block and 128 bit key AES cryptosystem has been presented in this paper. Optimized and Synthesizable VERILOG code is developed for the implementation of AES encrypt round and adapted for integrated AES encrypt/decrypt. The Sub Bytes/Inv Sub Bytes operations are implemented using composite field arithmetic in order to exploit the sub-pipelining. The AES algorithm implemented using Xilinx 13.2.

REFERENCES

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

[2] T. A. Pham, s. H. Mohammad and h. Yu, "area and power optimization for AES encryption module implementation on FPGA," in 18th international conference on automation and computing, pp. 1-6, September 2012.

[3] M, Anitha Christy, S.Sridevi Sathya Priya, N.M. Siva Mangai, "Design And Implementation Of Low Power Advanced Encryption Standard S-Box Using Pass Transistor Xor-And Logic, "In International Conference, Feb 2014.

[4] Edwin Nc Mui, "Practical Implementation Of Rijndael S-Box Using Combinational Logic", Custom R&D Engineer Texco Enterprise Pvt. Ltd.

[5] X. Zhang and K. K. Parhi, "High speed VLSI architectures for the AES algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 957-967, September 2004.

[6] S. K. Reddy S, R. Sakthivel and P. Praneeth, "VLSI implementation ofAES crypto processor for high throughput," *International Journal of Advanced Engineering Sciences and Technologies*, vol. 6, no. 1, pp.022-026, 2011.

AUTHORS' BIOGRAPAHY

Challa Vamshi Krishna, has completed his B.Tech in Electronics and Communication Engineering from VARADHA REDDY COLLEGE OF ENGINEERING, in 2011. He is pursuing his M.Tech in VLSI Design from SIDDHARTHA INSTITUTE OF ENGINEERING AND TECHNOLOGY, J.N.T.U.H affiliated college.

N. Shivakumar, is an Assistant Professor at Siddhartha Institute of Engineering and Technology, Hyderabad in ECE Department. He received his B.Tech degree in Electronics and Communication Engineering from Swarna Bharathi Institute of Science and Technology, Khammam and M.Tech degree in VLSI System Design from Narayana Engineering College, Hyderabad. He attended many workshops and conferences related to VLSI and Low power VLSI. His research interest is VLSI Technology and Design, communication systems and antennas.

Dr. D Subba Rao, is a proficient Ph.D person in the research area of Image Processing from Vel-Tech University, Chennai along with initial degrees of Bachelor of Technology in Electronics and Communication Engineering (ECE) from Dr. S G I E T, Markapur and Master of Technology in Embedded Systems from SRM University, Chennai. He has 13 years of teaching experience and has published 12 Papers in International Journals, 2 Papers in National Journals and has been noted under 4 International Conferences. He has a fellowship of The Institution of Electronics and Telecommunication Engineers (IETE) along with a Life time membership of Indian Society for Technical Education (ISTE). He is currently bounded as an Associate Professor and is being chaired as Head of the Department for Electronics and Communication Engineering discipline at Siddhartha Institute of Engineering and Technology, Ibrahimpatnam, Hyderabad.