FPGA Implementation of AES Algorithm

R. Sharadha, CH. Bhanu Prakash, M. J. C. Prasad

Abstract--This paper presents the FPGA implementation of AES algorithm Cryptography is the science of secret codes, enabling the confidentiality of communication through an insecure channel. It protects against unauthorized parties by preventing unauthorized alteration of use. Generally speaking, it uses a cryptographic system to transform a plaintext into a cipher text, using most of the time a key. To increase the computational speed parallelism and pipelining architecture have been implemented. The simulation is done using Xilinx 13.2 version.

Keywords – AES, FPGA, encryption, decryption, Rijndael, block cipher.

I. INTRODUCTION

The need for privacy has become a high priority for both governments and civilians desiring protection from signal and data interception. Widespread use of personal communications devices has only increased demand for a level of security on previously insecure communications. Confidentiality of network communications, for example, is of great importance for e-commerce and other network applications. However, the applications of cryptography go far beyond simple confidentiality. In particular, cryptography allows the network business and customer to verify the authenticity and integrity of their transactions. If the trend to a global electronic marketplace continues, better cryptographic techniques will have to be developed to protect business transactions. Sensitive information sent over an open network may be scrambled into a form that cannot be understood by a hacker or eavesdropper. This is done using a mathematical formula, known as an encryption algorithm, which transforms the bits of the message into an unintelligible form. The intended recipient has a decryption algorithm for extracting the original message. There are many examples of information on open networks, which need to be protected in this way, for instance, bank account details, credit card transactions, or confidential health or tax records.

II. DEFINATIONS

Cryptosystems can provide confidentiality, authenticity, integrity, and non repudiation services. It does not provide availability of data or systems

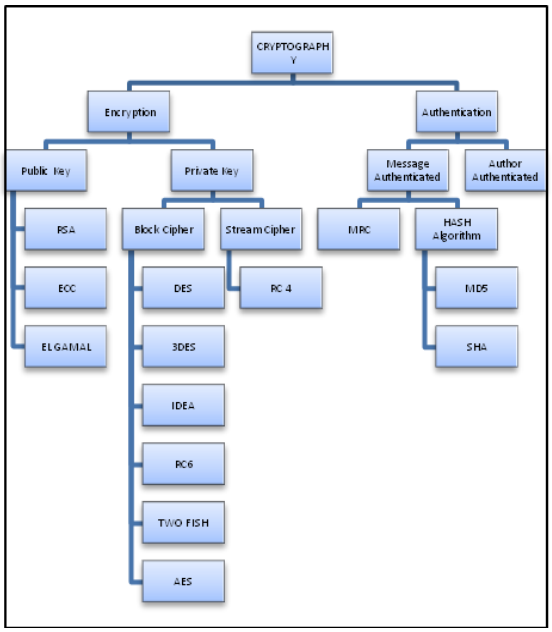


Fig 1 Basic Block diagram of cryptography

Confidentiality means that unauthorized parties cannot access information.

Authenticity refers to validating the source of the message to ensure the sender is properly identified.

Integrity provides assurance that the message was not modified during transmission, accidentally or intentionally.

Non repudiation means that a sender cannot deny sending the message at a later date, and the receiver cannot deny receiving it.

Cipher [3] is any method of encrypting text (concealing its readability and meaning). It is also sometimes used to refer to the encrypted text message itself. A *block cipher* is one that breaks a message up into chunks and combines a key with each chunk (for example, 64-bits of text). A *stream cipher* is one that applies a key to each bit, one at a time. Most modern ciphers are block ciphers.

III. ENCRYPTION CLASSES

There are two classes of algorithm in encryption, an asymmetric key and symmetric key.

Fig 2.Classification of cryptography

Manuscript received December, 2013.

R. Sharadha, M.Tech in Digital Systems & Computer Electronics at Mallareddy Engineering College, Secunderabad,India.

CH. Bhanu prakash, Assistant Professor in the department of Electronics and Communication Engineering, MREC, India.

Dr. M. J. C. Prasad, is presently working as a Head of the department of Electronics and Communication Engineering, MREC, Secunderabad, Andhra Pradesh, India.

A. Asymmetric Key or Public Key

In an asymmetric key algorithm, there are two keys. One must be public and it is used to encrypt the data. The other key is a private one and it is used to decrypt the information. This system is also used to sign a message digitally (Mao, 2003). Rivest- Shamir-Adleman (RSA) is widely used asymmetric key algorithm for decades and Elliptic Curve Cryptography (ECC) as an alternative to RSA which offers highest security with small bit length of key.

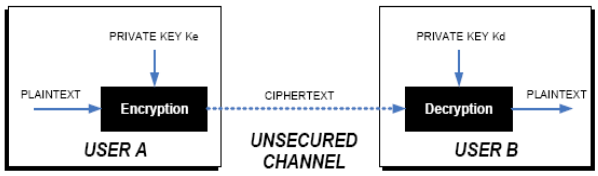


Fig 3 the asymmetry key cryptography

B. Symmetric Key or Private Key

In a symmetric or private key algorithm, in the ordinary case, the communication only uses only one key. Data Encryption Standard (DES) and CAST128 are example of symmetric key algorithm. Figure 4 shows the mechanism of private key cryptography.

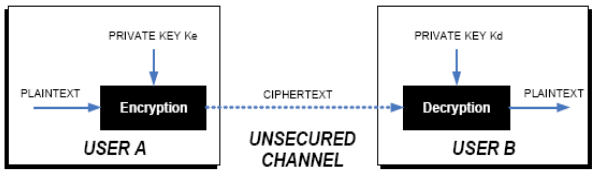


Fig 4. The symmetry key cryptography.

IV. TYPES OF CIPHER

pf cipher stream cipher and block cipher. Let us look at the j=key features of both and compare them.

A. Key Features of Stream Ciphers:

- Stream cipher treats the message as a stream of bits and performs mathematical functions on them individually.
- Operate on small units of plaintext, bits
- Symmetric encryption
- Usually implemented in hardware
- Encrypts by operating on a continuous data stream
- Some stream cipher use stream generator
- Statistically unpredictable
- Much faster than any block cipher
- Effective Stream algorithm contains
- Long period of no repeating patterns within key stream values
- Statistically unpredictable
- The key stream is not linearly related to the key

B. Key Features of Block Ciphers:

1. Operate on fixed size blocks of plain text
2. Breaks the plaintext into blocks and encrypts each with the same algorithm
3. Apply an identical encryption algorithm and key to each block
4. The properties of a cipher should contain confusion and diffusion
5. Spread the plaintext character over many cipher text characters. Done using permutations
6. Different unknown key values cause confusion

- Putting the bits within the plaintext through many functions cause diffusion
- Accomplished through p-boxes
- Conceals statistical connection using substitution
- Block cipher use S-boxes
- An S-box is non-linear because it generates a 4-bits output string from 6 bits input
- Are more suitable for software implementations, because they work with blocks of data which is usually the width of a data bus (64 bits).

C. Comparison

Stream ciphers are more difficult to implement correctly, and prone to weaknesses based on usage - since the principles are similar to one-time pad, the keystream has very strict requirements. On the other hand, that's usually the tricky part, and can be offloaded to e.g. and external box.

Stream ciphers do not provide integrity protection or authentication, whereas some block ciphers (depedning on mode) can provide integrity protection, in addition to confidentiality.

Stream ciphers are typically faster than block, but that has its own price.

V. ADVANCED ENCRYPTION STANDARD

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[2].

The Rijndael whose name is based on the names of its two Belgian inventors, Joan Daemen and Vincent Rijmen is a Block cipher, which means that it works on fixed-length group of bits, which are called blocks. AES uses three different key sizes: 128, 192 and 256 bits. the seliant feature of aes encryption and decryption are high throughput, parameter flexibilty, implentation flexibilty, no known security attack[2].

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. Both DES (Data Encryption Standard) and AES are defined as symmetric key block ciphers, with the main difference being the bit length of the key (56 bit for DES)[2].

Fig 2.1 shows a Structure of Advanced Encryption Standard In the fig 2.1 In the AES 128 we have 10 rounds of operations to be carried out. The first round consists of all the five operation like Preround operation, subbyte, shift rows, mix columns and Add round key operations. From 2nd round to 9th round have four operations subbyte, shift rows, mix columns and Add round key operations. And the last 10th round consists of three operations sub byte, shift rows and Add round key operations [1].

In AES 128 the process of generating the key is each round key is a 128-bit array generated as,

- Input key of 128 bit is divided into four parts of 32 bits as columns in a matrix4*4.
- 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 it gives
- The above output is taken as 0th column of the generated new key.

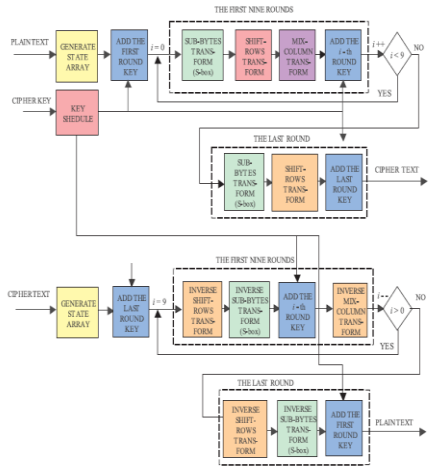


Fig 4 Structure of Advanced Encryption Standard

- .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.

In this way we generate new keys of 128 bits in AES 128 algorithm by attaching the four obtained columns of new key.

VI. STANDARD AES ALGORITHM SPECIFICATIONS

For the AES algorithm**,** the length of the input block, the output block and the State is 128 bits. This is words (number of columns) in the State.

For the AES algorithm**,** the length of the Cipher Key, *K*, is 128, 192, or 256 bits**.** The key length is represented by N_k = 4, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key.

For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr =10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8.

The only Key-Block-Round combinations that conform to this standard are given in Table 1

Table 1 key-Block-Round Combinations

AES Input data *(in bits)*	Key Length *(Nw words)*	Block size *(Nb words)*	Number of Rounds *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations[2][1]:

1) Byte substitution using a substitution table (S-box)
2) Shifting rows of the State array by different offsets
3) Mixing the data within each column of the State array
4) Adding a Round Key to the State

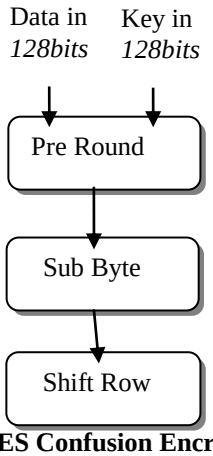


Fig 5 AES Confusion Encryption algorithm

AES is an iterated block cipher with a fixed block size of 128 and a variable key length. The different transformations operate on the intermediate results, called *state*. The state is a rectangular array of bytes and since the block size is 128 bits, which is 16 bytes, the rectangular array is of dimensions 4x4. The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key, denoted Nk, is equal to the key length divided by 32[1].

It is very *important* to know that the cipher input bytes are mapped onto the state bytes in the order a0,0, a1,0, a2,0, a3,0, a0,1, a1,1, a2,1, a3,1 ... and the bytes of the cipher0 key are mapped onto the array in the order k0,0, k1,0, k2,0, k3,0, k0,1, k1,1, k2,1, k3,1 ... At the end of the cipher operation, the cipher output is extracted from the state by taking the state bytes in the same order.

A state: A key:

a0,0	a0,1	a0,2	a0,3		k0,0	k0,1	k0,2	k0,3
a1,0	a1,1	a1,2	a1,3		k1,0	k1,1	k1,2	k1,3
a2,0	a2,1	a2,2	a2,3		k2,0	k2,1	k2,2	k2,3
a3,0	a3,1	a3,2	a3,3		k3,0	k3,1	k3,2	k3,3

AES uses a variable number of rounds, which are fixed: A key of size 128 has 10 rounds. A key of size 192 has 12 rounds. A key of size 256 has 14 rounds.

During each round, the following operations are applied on the state:

1. Sub Bytes: Every byte in the state is replaced by another one, using the Rijndael S-Box

2. Shift Row: Every row in the 4x4 array is shifted a certain amount to the left

Note: In the final round, the Mix Column operation is omitted.

A. Pre Round Operation

In this operation, [1] a given data input (128 bits) is bitwise XORed with User defined Key (128 bits) to generate a cipher text of 128bits as shown in table 2.

Where: b (i,j) = a(i,j) XOR k(i,j)

A graphical representation of this operation can be seen in fig 6

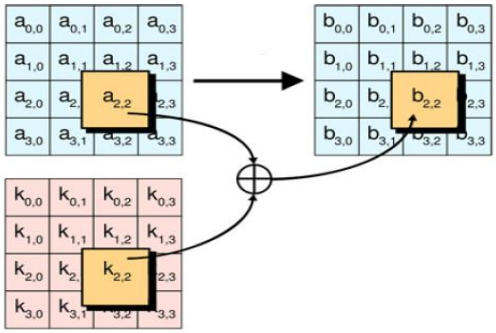


Fig 6 AES encryption Pre round Operation

Table 2 pre round operation

a0,0	a0,1	a0,2	a0,3		k0,0	k0,1	k0,2	k0,3		b0,0	b0,1	b0,2	b0,3
a1,0	a1,1	a1,2	a1,3	XOR	k2,0	k2,1	k2,2	k2,3	=	b2,0	b2,1	b2,2	b2,3
a2,0	a2,1	a2,2	a2,3		k1,0	k1,1	k1,2	k1,3		b1,0	b1,1	b1,2	b1,3
a3,0	a3,1	a3,2	a3,3		k3,0	k3,1	k3,2	k3,3		b3,0	b3,1	b3,2	b3,3

B. Sub Byte Transformation

The SubBytes operation is a non-linear byte substitution, operating on each byte of the state independently. The substitution table (S-Box) is invertible and is constructed by the composition of two transformations:

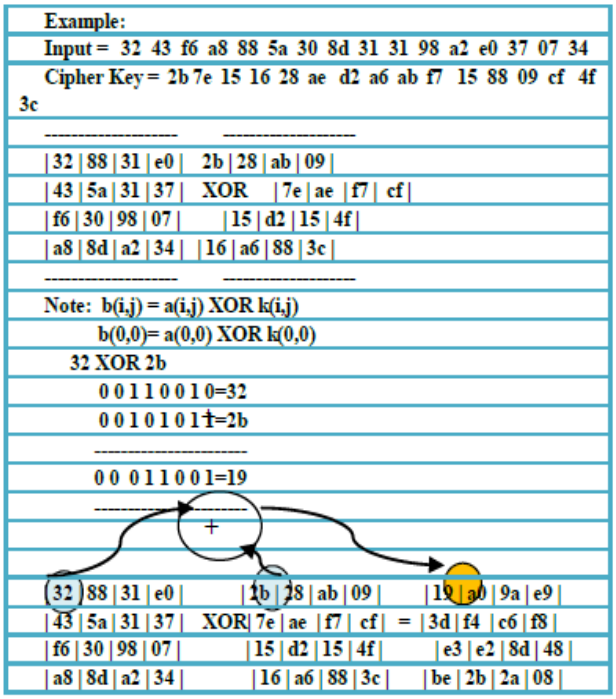
1. Take the multiplicative inverse in Rijndael's finite field
2. Apply an affine transformation as described below

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

For $0 < i < 8$, where b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte *c* with the value {63} or {01100011}

In matrix form, the affine transformation element of the S-box can be expressed asShown in figure 5

Table 3 Example of pre-round operation



$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Fig 5 Affine Transformation of S-BOX

Figure 6 illustrates the effect of the **SubBytes()** transformation on the State

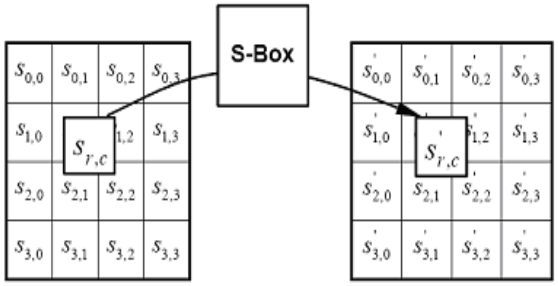


Fig 6 SubBytes() applies the S-Box to each byte of the state

The S-box used in the **SubBytes()** transformation is presented in hexadecimal form. As shown in table 2.2.

Table 4 Substitution values for the bytes xy (in hexadecimal format

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Since the S-Box is independent of any input, pre-calculated forms are used, if enough memory (256 bytes for one S-Box) is available [1]. Each byte of the state is then substituted by the value in the S-Box whose index corresponds to the value in the state: a (i,j) = SBox[a(i,j)].

Since the S-Box is independent of any input, pre-calculated forms are used, if enough memory (256 bytes for one S-Box) is available [1].

Each byte of the state is then substituted by the value in the S-Box whose index corresponds to the value in the state: a (i,j) = SBox[a(i,j)].

Example:

Pre Round O/p:

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

S-BOX: 0th Row and 8th Column

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	30

Sub Byte Output:

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

S-BOX: ith Row and ith Column

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

Pre Round S-box Transformation Sub Byte Output

Figure 7 example of pre-round and sub byte outpout.

C. Shift Row Operation

In the ShiftRows() transformation, as shown in fig 8, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets)[1][3][4]. The first row, $r = 0$, is not shifted. Specifically, the ShiftRows() transformation proceeds as follows:

$S'_{r,c}=S_{r,(c+shift(r,Nb))modNb}$ for $0<r<4$ and $a<=c<Nb$

where the shift value *shift*(*r*,*Nb*) depends on the row number, *r*, as follows (recall that $Nb = 4$):

shift(1,4) =1; *shift*(2,4) = 2 ; *shift*(3,4) = 3

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.

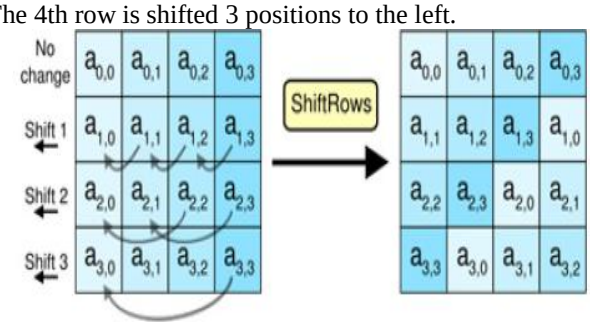


Fig 8 AES Encryption Shift Row Operation

Example:

SubByte Output is given as an input to *ShiftRow* Operation.

SubByte o/p

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

Shift Row: No change; Shift left1; Shift left2; Shift left3

Shift Row Output

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

VII. MIXCOLUMNS () TRANSFORMATION

The MixColumns() transformation operates on the State column-by-column, treating each column as a four-term polynomial [1][4]. The columns are considered as polynomials over GF(2^8) and multiplied modulo $x4 + 1$ with a fixed polynomial $a(x)$, given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

Fig 8 result of the multiplication

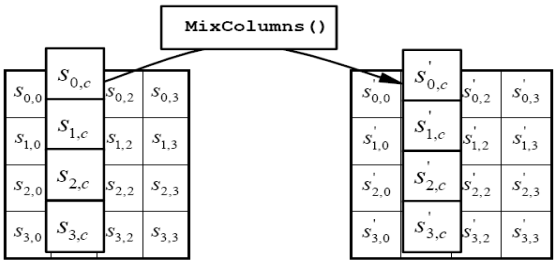


Fig 9 Mixcolumn() Transformation

VIII. ADD ROUND KEY TRANSFORMATION

In the Add Round Key() transformation, a Round Key is added to the State by a simple bitwise XOR operation[1]. Each Round Key consists of *Nb* words from the key schedule (described in Sec.5.2). Those *Nb* words are each added into the columns of the State, such that

$[S`_{0,c}, S`_{1,c}, S`_{2,c}, S`_{3,c}] = [S_{0,c}, S_{1,c}, S_{2,c}]$ (xor) $[W_{round*Nb+c}]$ for $0 <= c < Nb$,

where [*wi*] are the key schedule words described in Sec. 3.5 and *round* is a value in the range 0 ε *round* ε *Nr*

A. *Key Schedule*

The Round Keys are derived from the Cipher Key by means of the key schedule[3]. This consists of two components: the Key Expansion and the Round Key Selection

B. *Key Expansion*

The Key Expansion generates a total of $\boldsymbol{N_b}$ ($\boldsymbol{Nr}$ + 1) words: the algorithm requires an initial set of $\boldsymbol{N_b}$ words, and each of the ***Nr*** rounds requires ***Nb*** words of key data [3]. The resulting key schedule consists of a linear array of 4-byte words, denoted [*wi*], with *i* in the range 0 ε *i* < ***Nb***(***Nr*** + 1).

It is important to note that the Key Expansion routine for 256-bit Cipher Keys (***Nk*** = 8) is slightly different than for 128- and 192-bit Cipher Keys. If ***Nk*** = 8 and i-4 is a multiple of ***Nk***, then **SubWord()** is applied to **w**[**i-1**] prior to the XOR.

C. *Round Key Selection*

Round key i is given by the Round Key buffer words W[Nb*i]to W[Nb*(i+1)].

IX. DESCRIPTION OF AES DECRYPTION ALGORITHM

The AES Encryption algorithm can be inverted end then implemented in reverse order to produce a straightforward Inverse AES algorithm to decrypt the AES encrypted data.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

The individual transformations used in the Decryptor are - *InvShiftRows(), InvSubBytes(),Inv MixColumns(), and AddRoundKey()[5][6].*

During each round, the following operations are applied on the state:

7. **Inv Sub Bytes**: Every byte in the state is replaced by another one, using the Rijndeal Inv S-Box

2. **Inv Shift Row**: Every row in the 4x4 array is shifted a certain amount to the right

3. **Inv Mix Column**: A linear transformation on the columns of the state

4. **Add Round Key**: Each byte of the state is combined with a round key, which is a different key for each round and derived from the Rijndeal ke e

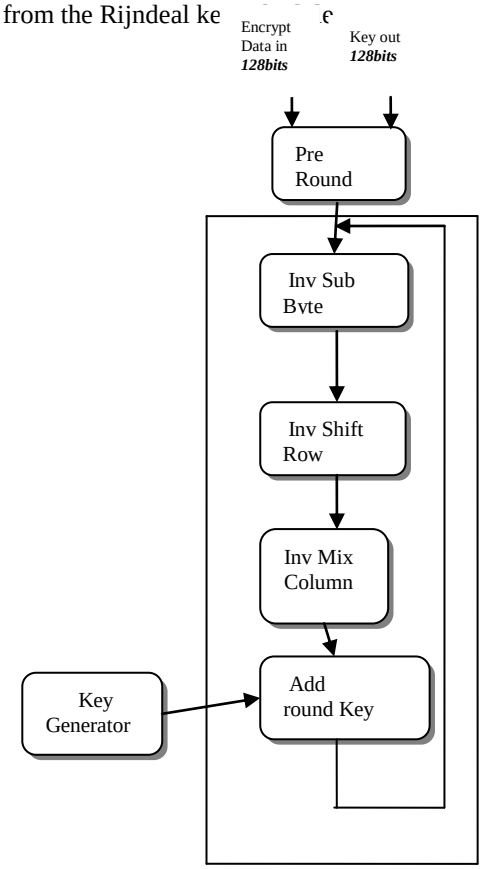


Figure 10 AES Decryption algorithm

X. SIMULATION RESULTS OF AES ALGORITHM

The simulation results of AES Algorithm consists of 4 internal operational architecture (Sub Byte, Shift Row, Mix Column and Add Round key) and one externally generated key using Key generator Hardware.

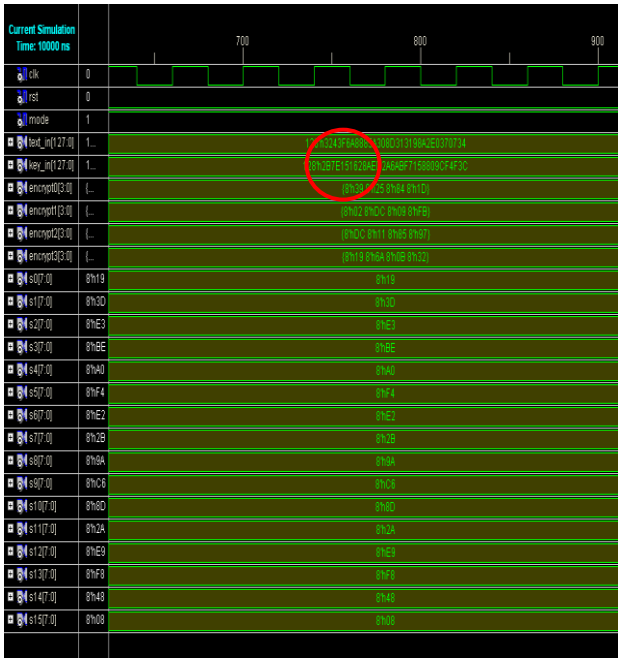
A. Pre Round Simulation Results:

THEORITCAL OUTPUT:

Pre round o/p

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

Simulated Output:



B.Sub Byte Simulation Results:

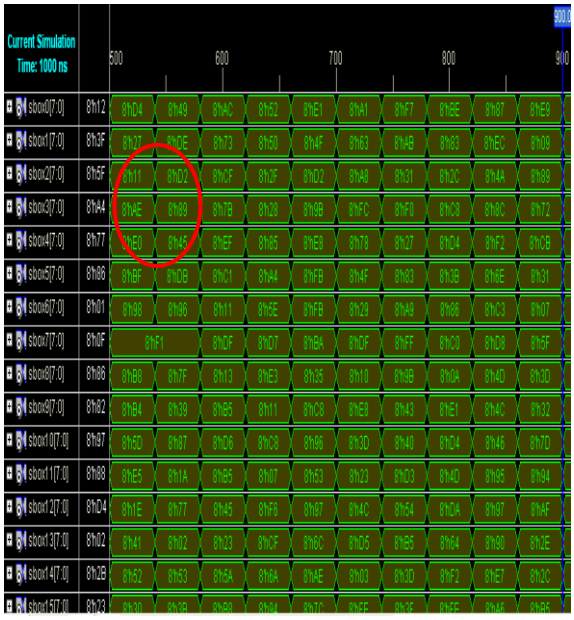
THEORITCAL OUTPUT:

Sub Byte Output: (Refer Sec 2.5.3)

---

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

Simulated Output:



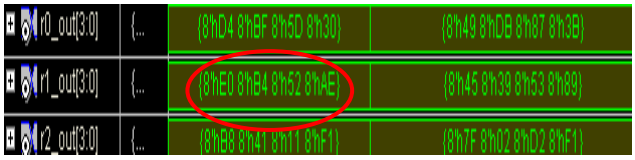
C. Shift Row Simulation Results:

THEORITCAL OUTPUT:

Shift Row Output (Refer Sec 2.4.3)

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

Simulated Output:



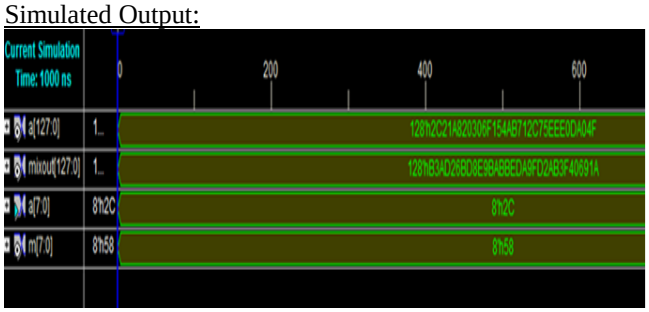
D. Mixcolums Simulation Results

THEORITCAL OUTPUT:

Mix columns Output (Refer Sec 2.5.4)

b3	8e	da	3f
ad	9b	9f	40
26	ab	d2	69
bd	be	ab	1a

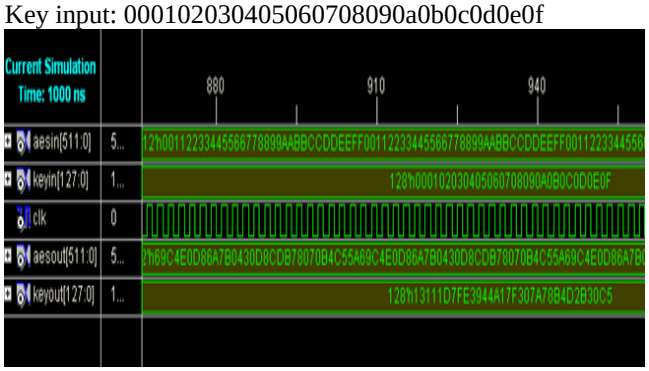
Simulated Output:



E. Encryption Simulation Output

AES input: 00112233445566778899aabbccddeeff

Key input: 000102030405060708090a0b0c0d0e0f



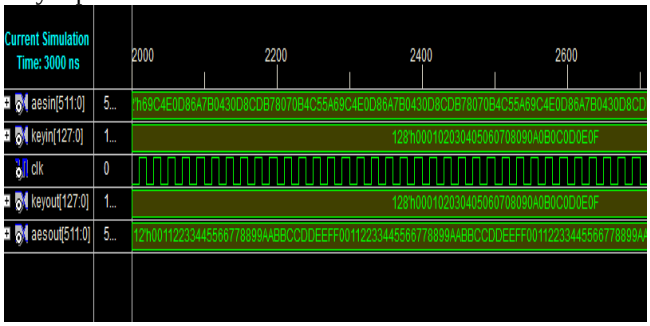
AES output: 69c4e0d86a7b0d8cdb78070b4c55a

Key output: 13111d7fe3944a17f307a78b4d2b30c5

F. Decryption Simulation Output

Aes input: 69c4e0d86a7b0d8cdb78070b4c55a

Key input: 000102030405060708090a0b0c0d0e0f



Aes output: 00112233445566778899aabbccddeeff

Key output: 000102030405060708090a0b0c0d0e0f

XI. CONCLUSION

A new AES VLSI architecture is developed to reduce the cost using a minimalist bit serial approach. The critical paths in the hardware implementation have shown to be SubBytes, MixColumns and all the operations are integrated into a simple architecture in which all the operations are concurrently done by using the same blocks using different control signals, which is very important develop an architecture to minimize the cost of the implementation (i.e. gate count).By using a true low level bit-serial approach, minimum cost AES co-processor architecture has been achieved. This architecture can be used in many military, industrial, and commercial applications that require compactness and low cost.

ACKNOWLEDGMENT

I, R. Sharadha highly indebted to Head of the department of Electronics and Communication Engineering. Dr. M. J. C. Prasad. CH. Bhanu Prakash. Assistant Professor in the department of Electronics and Communication Engineering, MREC for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

REFERENCES

1. WANG Wei, CHEN Jie, XU Fei3 “An Implementation of AES Algorithm Based on FPGA”, 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012)
2. Deshpande, A.M. Deshpande, M.S. Kayatanavar, D.N. “FPGA implementation of AES encryption and decryption”, IEEE Transactions, Print ISBN: 978-1-4244-4789-3, Jun 2009.
3. Muhammad H. Rais and Syed M. Qasim “Efficient Hardware Realization of Advanced Encryption Standard Algorithm using FPGA”, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.9, September 2009
4. Marko Mali, Franc Novak and Anton Biasizzo “Hardware Implementation Of AES Algorithm”, Journal of Electrical Engineering, VOL. 56, NO. 9-10, 2005, 265–269
5. Rajender Manteena, “A VHDL Implemetation of the Advanced Encryption Standard-Rijndael Algorithm”, College of Engineering University of South Florida, 2004.
6. S. Morioka and A. Satoh “A 10-Gbps Full AES-CryptoDesign with a Twisted BDD S-Box Architecture”, *IEEE Transactions on VLSI Systems*, Vol. 12, No. 7, July 2004, pp.686-691.

AUTHORS PROFILE

R. Sharadha is presently pursuing final semester M.Tech in Digital Systems & Computer Electronics at Mallareddy Engineering College, Secunderabad. She received degree B.Tech in Electronics and communication from G. Narayanamma Institute of Technology and science. Her areas of interest are Digital system designs, VLSI.

CH. Bhanu prakash is presently working as an Assistant Professor in the department of Electronics and Communication Engineering, MREC, Secunderabad, Andhra Pradesh, India. He is having 5Years of teaching experience. His areas of interest are VLSI and Embedded systems.

Dr. M. J. C. Prasad is presently working as a Head of the department of Electronics and Communication Engineering, MREC, Secunderabad, Andhra Pradesh, India. He is having 15 years of teaching experience. His areas of interest are Communication systems, Digital Systems, Image Processing. Digital signal processing, Advance DSP Systems.