Perspectives in Communication, Embedded-Systems and Signal-Processing (PiCES)
ISSN: 2566-932X, Vol. 2, Issue 2, May 2018

VLSI implementation of AES Encryption/Decryption Algorithm using FPGA

Nayana G H[1], Karthik K[2], Mahalakshmi T S[3], Manasa H R[4]

1. Electronics and Communication Department, New Horizon College of Engineering, Bengaluru, India, nayanagh.0109@gmail.com
2. Electronics and Communication Department, New Horizon College of Engineering, Bengaluru, India, karthikroy1100@gmail.com
3. Electronics and Communication Department, New Horizon College of Engineering, Bengaluru, India, pammemaha@gmail.com
4. Electronics and Communication Department, New Horizon College of Engineering, Bengaluru, India, manasareddy148@gmail.com

Abstract: AES stands for Advanced Encryption Standard and it is very efficient algorithm available today. Depending upon the key length it will become more efficient, 3 key length options supported here, viz. 128,192 bit and 256 bit key length. We generate encrypted data from the raw data bits and the same encrypted data is passed to the decryption blocks to recover the data back. From the hardware perspective, Field Programmable Gate Array (FPGA) is being used for hardware implementations in encryption process. The design of AES will be carried out in MATLAB and verified. RTL coding done using Verilog HDL, Xilinx Spartan 3A FPGA will be used in implementation. Modelsim is used in simulation Function and Timing simulations. The design will be implemented as per VLSI Industry Standard FPGA flow.

Keywords: AES, NIST, FPGA, MATLAB, Verilog HDL.

I. INTRODUCTION

AES is solitary of the algorithms with the purpose of being used intended for security, which is proposed under the NIST. AES perform the encryption of the data as an ECB cipher type. AES perform encryption of 128 bit data, which is symmetric (that is same key is used for encryption as well as the decryption process). It has four modules, at the end of every round a secret message text is obtained as product.

II. AES CIPHER

ADVANCED ENCRYPTION STANDARD, the AES is a subset of a much larger encryption algorithm known as Rijndael, which was one of many proposals to the NIST challenging for becoming a standard encryption algorithm. On October of 2000, the NIST announced the Rijndael algorithm as the winner due to the best overall keep count in security, performance, efficiency, functioning capability and simplicity. The AES algorithm is a symmetric cipher. In addition, the AES algorithm is a block cipher as it operate on fixed-length groups of bits (blocks). The AES algorithm operates on blocks of 128 bits, by using cipher keys with lengths of 128, 192 or 256 bits for the encryption process. Although the original Rijndael encryption algorithm was capable of processing different blocks sizes as well as using several other cipher key lengths, but the NIST did not adopt these additional features in the AES. The advance encryption standard is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of application. The 128-bit plain text and 128-bit initial key, as well as the 128-bit output of cipher text, are all divided into four 32-bit consecutive units respectively controlled by the clock. The Work focuses on 4 modules namely,

1. Add round key.
2. Substitution Bytes.
3. Shift Rows.
4. Mix Columns.

In this paper we propose AES-128 architecture, in which we have 128 bit input plain text and obtain 128 bit output cipher text.

III. AES PROCESS

AES is based on a design principle known as a substitution–permutation network, a combination of both substitution and permutation, and is fast in both software and hardware. AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.

AES operate on a 4×4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger chunk size and have additional columns in the state. Most AES calculation are done in a particular finite field. The key size used for an AES cipher specifies the number of repetitions of conversion rounds that convert the input, called the plaintext, into the final output, called the cipher text. The numbers of cycles of recurrence are as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform cipher text back into the original plaintext using the same encryption key.

AES mainly consists of 4 stages in its encryption as well as decryption process, the stages are

Add round key, substitution byte, shift rows, mix column. For the decryption process the stages remains same but are inverse.

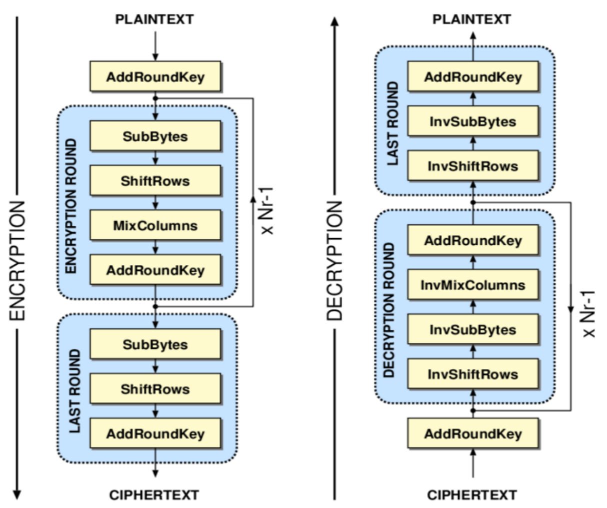


Fig 1. AES process

A. Substitution Bytes

In the SubBytes step, each byte in the state matrix is replaced with a SubByte using an 8-bit substitution box, the S-box. This operation provides the non-linearity in the cipher. While performing the encryption, the sbox is used. While performing the decryption,the invSubBytes step (the inverse of SubBytes) is used.

Consider FD BC is your data, then Fth row and Dth column corresponding value is replaces FD, similarly BC is also done.

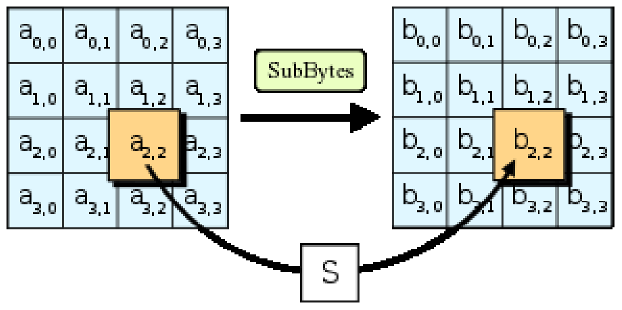
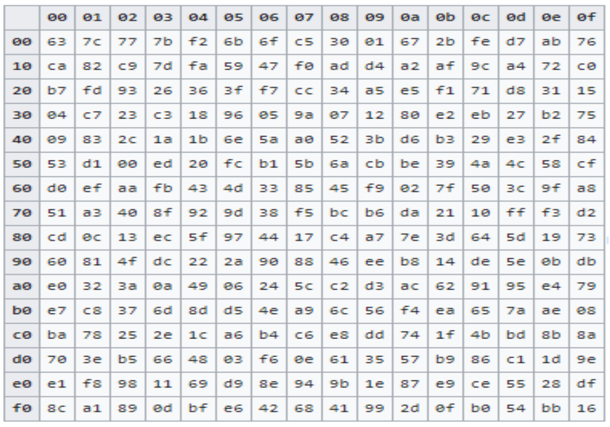

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S

Fig 2. Substitution of Bytes

Therefore from the sbox we notice that FD corresponds to 54, thus FD is replaced with 54 in the further steps. Similarly BC corresponds to 65.

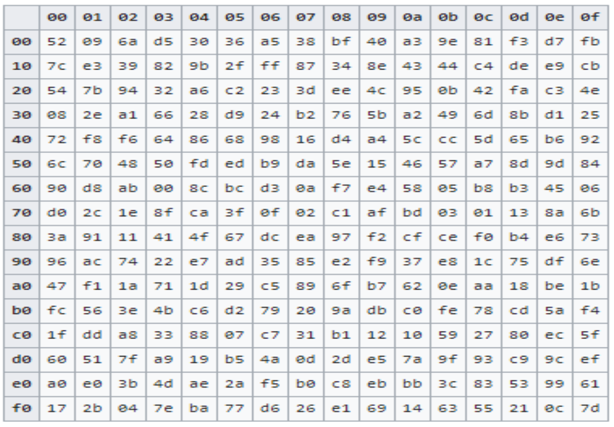
B. Shift Rows

The Shift Rows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. The importance of this step is to avoid the columns being encrypted independently, in which case AES degenerates into four independent block ciphers.



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

Fig 3. S Box



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

Fig 4. Inverse S Box

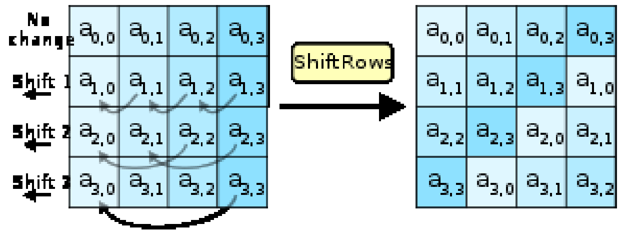

No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Fig 5. Shift Rows

The below shows an example of the shift rows operation,

$$\begin{vmatrix} 87 & F2 & 4D & 97 \\ EC & 6E & 4C & 90 \\ 4A & C3 & 46 & E7 \\ 8C & D8 & 95 & A6 \end{vmatrix} \Rightarrow \begin{vmatrix} 87 & F2 & 4D & 97 \\ 6E & 4C & 90 & EC \\ 46 & E7 & 4A & C3 \\ A6 & 8C & D8 & 95 \end{vmatrix}$$

C. Mix Column

In the Mix Columns step, the four bytes of each column of the state are combined using an invertible linear transformation. The Mix Columns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with Shift Rows, Mix Columns provides diffusion in the cipher.

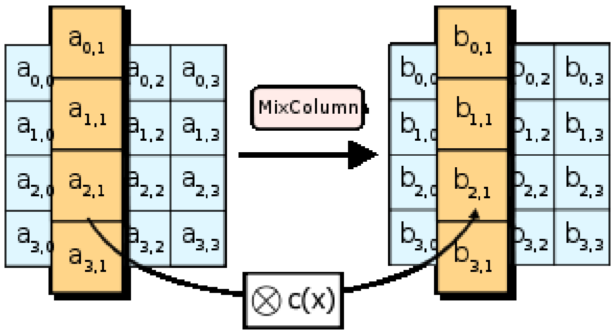


Fig 6. Mix Column

During this operation, each column is transformed using a fixed matrix (matrix left-multiplied by column gives new value of column in the state), the matrix involved in the mix column step during the encryption is given below,

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \qquad ....(2)$$

Matrix multiplication is composed of multiplication and addition of the entries. Entries are 8 bit bytes treated as coefficients of polynomial of order. Addition is simply XOR. Multiplication is modulo irreducible polynomial. The matrix that is involved in the decryption process is given below,

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \qquad ....(3)$$

The operations performed can be shown clearly in the below equations, where Fig 6 corresponds to the encryption process and Fig 7 corresponds to the decryption process.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Fig 7. Encryption process in Mix column.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Fig 8. Decryption process in Mix column.

D. *Add Round Key*

In the Add Round Key step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule, each subkey is of same size as that of the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR operation. The operation is as shown in below Fig 8.

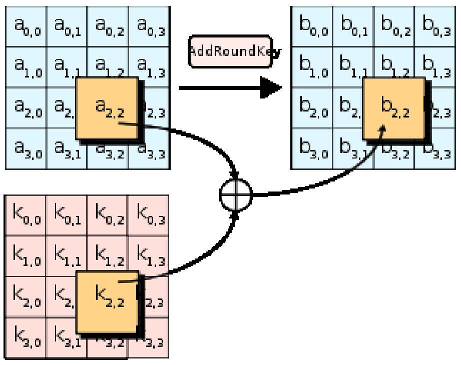


Fig 9. Add Round Key module.

IV. PROPOSED METHOD

Perviously in the mixed column module we used a separate key for multiplication upon which we get the result, this becomes very time consuming thus new method is proposed to over come that. Here in this method we make use of lograthmic and anti-lograthmic tables namely ltable and etable.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01

Fig 10. ETable

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	23	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	07

Fig 11. LTable

Upon receiving the output from the previous step of mix column, that is shift rows results in case of encryption and the cipher text in case of decryption processes, the values are checked in the ltable same as done in case of sbox. After we obtain the ltable values for corresponding values we add those two values, here add refers to xor operation i.e modulo 2 additions. The result thus obtained is checked in the corresponding row n column of the e table and thus replaces the result value with the etable value. Now if this result should be of 8bit in length, else if it is 9bit value then it is Xored with 11b to obtain 8bit value. There is a chance that the result may exceed upto maximum of 13bit range.

If the result is a 10 bit value then the 11b is left shifted by 1 bit and the it is xored with the 10 bit value result until 8 bit value is obtained. Similarly if 11 bit, the 11b is left shifted by 2 bits, if 12 bits 11b is left shifted by 3 bits, if 13 bits the 11b is left shifted by 4 bits, and is xored with the result untill a 8 bit value is obtained.

V. COMPARISION

Table 1. Comparison of area occupied by mix columns module

	Available Area	**Area used**
Existing Mix Column method.	14752 units	10664 units that is almost 72%
Proposed Mix Column method.	14752 units	134 units that is almost 0.009%

Table 2. Time consumed comparison

	Time consumed for single round (nano seconds).
Existing Mix Column method.	26.960ns
Proposed Mix Column method.	7.232ns

From the table 1 and table 2 it is clear that mix columns module that is been proposed in this paper occupies very less area and less time compared to the existing method. Thus the proposed method is best compared to existing.

VI. CONCLUSION

Implementation of AES-128 technique is targeted on this FPGA device to provide physical security and optimization of these three factors. Hardware Description Language (HDL) is used to define the AES architecture on FPGA device. Once the programming is fused to FPGA, no one can modify the content present in it and hence tampering of code and information is not possible. The main objective of this research is to design the AES structure with less area, delay and power consumption. The compact composite S-Box structures for AES 128 bits technique is designed to provide an improved S-Box with the reasonable reduction in area, delay and power. Further, the design of MixColumn and Inverse MixColumn architecture is concentrated with the introduction of reduced Xtime structures in order to reduce the area and power factors. Incorporation of these two techniques in the design of FPGA based AES results a high performance AES 128 bit system with improvement in the area, delay and power reduction.

VII. FUTURE SCOPE

There are many possible extensions of this work that can be undertaken as future research work cum projects. The proposed AES-128 bit technique comprising of compact composite and reduced Xtime based Inverse Mix Column can be implemented on a small chip size, around 65 nm (CMOS technology based) or less in order to satisfy the needs of applications such as smart cards and RFID tags. The isomorphic mapping process of composite S-Box can be concentrated in future for further improvement in the system performance with the reduction of area and power utilization. The proposed AES technique is designed in consideration with 128 bit key size. This key size can be increased to 192 bits and 256 bits as per the specifications. According to the proposed technique, the key size is fixed. In future, a variable key size for AES can be designed in order to secure the data with different types of cipher forms. The delay reduction and speed increase algorithm presented in this research work suggests the practical way of producing this simulated algorithm into the commercial market after successful implementation on hardware device integrated with other interfacing and 135 intercommunicating devices. This process is considered to be the next level of implementation. A high performance FPGA device like Virtex (Ultra SCALE) can be used to improve the system performance with high speed computation process.

REFERENCES

[1] National Institute of Standards and Technology, "Announcing the Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, pp. 1-47, 26 November 2001.

[2] J.Daemenand, V.Rijmen, "AES Proposal: Rijndael", Document version 2, pp. 1-45, 3 September 1999, http://csrc.nist.gov/cryptotoolkit/AES/Rijndael/Rijndael.pdf

[3] M. M. WONG, M.L.D. Wong, "A high throughput low power compact AES s-box implementation using composite field arithmetic and algebraic form representation", proc. IEEE 2nd Aseasymposium on quality electronic design, pp 318-323, 2010.

[4] Monica libertori, Fernando Otero, J.C.Bonadero, Jorge Castineira, "AES-128 Cipher High Speed, Low Cost FPGA Implementation", 3rd Southern Conference on Programmable Logic, Institute of Electrical and Electronics Engineers xplore, ISBN:1-4244-0606-4, pp.195- 198, April 2007.

[5] Leelavathi.G, Prakasha S, Shaila K, Venugopal K R, L M Patnaik, "Design and Implementation of Advanced Encryption Algorithm with FPGA and ASIC", International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 3, ISSN: 2320 – 8791, pp. 1-8, June-July, 2013. 6. M.Matsui, "Linear cryptanalysis method for DES cipher", Springer, Eurocrypt, Lncs765, pp.386-397, 1994.

[6] Samir Palnitkar, "Verilog HDL- A Guide to Digital Design and Synthesis", Prentice Hall, pp. 3-10, 2003.

[7] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 197", 2001

[8] Joan Daemen and Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer, 2002. ISBN 3-540-42580-2 (2002).