ISSN 2320-9984

Volume 2 Issue 2 | June 2014 ISSN: 2320-9984 (Online)

International Journal of Modern Engineering & Management Research

Website: www.ijmemr.org

Design and Verification of Optimized Speed Advance Encryption Standard Method using HDL

Mradul Upadhyay
Research Scholar
Embedded System and VLSI Design
Gyan Ganga Institute of Technology and Sciences,
Jabalpur (M.P.) [INDIA]
Email: mradul_upadhyay1486@yahoo.com

Prof. Utsav Malviya
Assistant Professor
Electronics & Communication
*Gyan Ganga Institute of Technology and Sciences, *
Jabalpur (M.P.) [INDIA]
Email: utsavmalviya@ggits.org

Abstract—*In this paper we have proposed high throughput by swapping the AES algorithm internal stages in this proposed work shift row structure is operated before sub bytes (substitution bytes). In this proposed operation the AES encryption operation will not effect, with this process is streamlines the processes a 4 block of data rather then 16 block. The advantage of this is we can save area. This process repeats for 10 cycles and with the help of this we can encrypt 128 bits data with higher throughput. We have evaluated this performance of higher throughput and hardware area in Xilinx's SPARTAN-3E FPGA Family.*

1. INTRODUCTION

Advanced Encryption Standard (AES) is used for security purpose in Military application. All Cryptographic algorithm are used for security services for various application all the encryption technique are used in government and secret military communication.

Before AES we use DES(Data Encryption Standard) in the DES technique we can encrypt only 64-bit data and this data encryption standard is following by systematic algorithm called encryption algorithm. The AES algorithm is capable of using cryptographic keys of 128,192,256 bits to encrypt or decrypt the data.

Table 1: Rounds of AES

	Block Size Nb Word	Key Length Nk Words	No. of Round Nr
AES-128 bit key	4	4	10
AES-192 bit key	4	6	12
AES-256 bit key	4	8	16

Formula : [Round Nr = 6+max{N_b, N_k}]

N_b = 32 bit word in the block

N_k = 32 bit word in key

We can calculate the number of cycle (Rounds) taken to encrypt any data from this formula. The total number of rounds can be calculate with 6 necessary round and addition of Max (bit word in the block, bit word in the key). For example if we take 4 block size and 4 key length than it contains AES-128 bit key (32 * 4 = 128) because each key length contains 32 bit word key.

2. AES ALGORITHM

The AES used for encrypt and decrypt 128 bit plain text block. To encrypt this plain text we required 3 modes:128 bit, 192 bit 256 bit. Each has corresponding number of round. To encrypt the data we required 128 bit matrix and each row contains four bytes of group. The 4*4 Matrix is given below:

$$X = \begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{bmatrix}$$

Figure 1: Byte Matrix

The AES algorithm consists of four different simple operations.

- Byte substitution (Sub Bytes)
- Shift Row
- Mix column
- Add Round Key

Sub Bytes: All bytes are processed separately and it is a non linear byte substitution. Sub byte is invertible and constructed by the composition of following two transitions;

Inversion in the GF(2^8) field and modulo an irreducible polynomial is given by:

$$M(x) = x^8 + x^4 + x^3 + x + 1$$

Affine transformation define by as following :

$$Y = AX^{-1} + b$$

Where A is 8*8 fixed matrix and b is 8*1 vector matrix.

Substitution byte stage is completed with the help of S-box. S-box is fixed and it is nothing but a matrix.

We will see how this matrix is affected in each round. The particular value in the S-box can be determined by breaking the bytes into nibble. The left most nibble of the byte is specify the particular row of the S-box and the right most nibble of the S-box is specifies the column. For example the byte {19} select row 1, column 9 which contains the value {d4}. This value is used to update the state matrix.

We can understand this process with the help of the diagram given below:

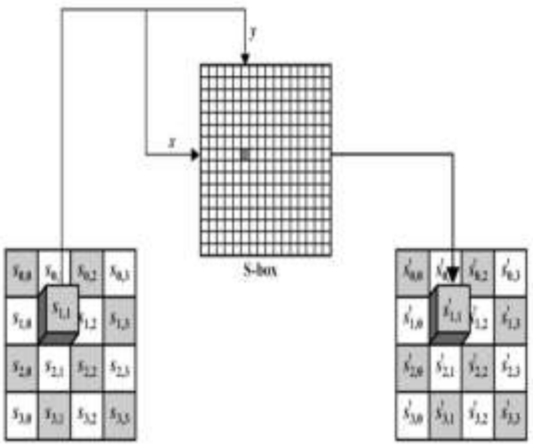
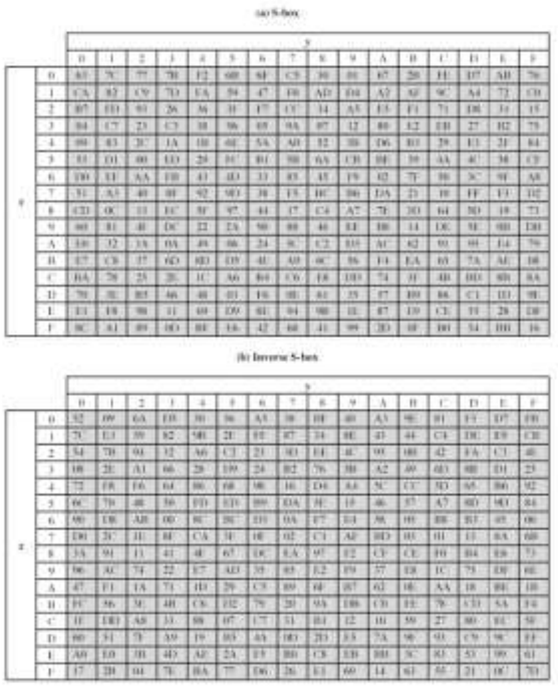


Figure 2 : S-Box creation

On the other hand the inverse sub byte transformation (known as InvSubByte) makes use of an inverse S-box. In this case we select the value {d4} and get the value {19}.This can be verify with the help of diagram given below.



(a) S-box

	y															
x	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

(b) Inverse S-box

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

Figure 3: S-Box

Shift Row: In this stage

- The first row is not change.
- The second row is circular shifted by 1 byte to the left.
- The third row is circular shifted by 2 byte to the left.
- The fourth row is circular shifted by 3 byte to the left.

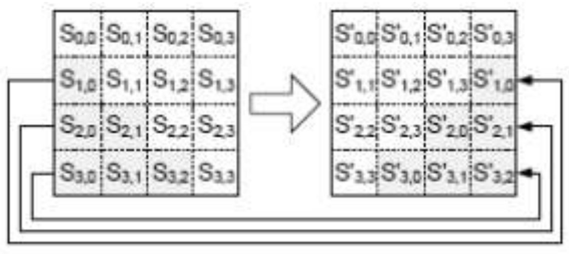


Figure 4: Shift Row

The inverse Shift Row transformation performs this shifting operation in the opposite (right Shift) direction.

Mix column: the mix column transformation operates column by column and these column will be consider as a four term polynomial. The column are consider as four term polynomial over GF(2^8) are multiplied x^4+1 with the fixed polynomial a(x) is given by:

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

We can see this operation in the form of matrix as:

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

Figure 5: Mix Column

Where $0 \le j \le 3$.

The result of this matrix multiplication is given by:

$$S'_{0,j} = (02 \cdot S_{0,j}) \oplus (03 \cdot S_{1,j}) \oplus S_{2,j} \oplus S_{3,j}$$
$$S'_{1,j} = S_{0,j} \oplus (02 \cdot S_{1,j}) \oplus (03 \cdot S_{2,j}) \oplus S_{3,j}$$
$$S'_{2,j} = S_{0,j} \oplus S_{1,j} \oplus (02 \cdot S_{2,j}) \oplus (03 \cdot S_{3,j})$$
$$S'_{3,j} = (03 \cdot S_{0,j}) \oplus S_{1,j} \oplus S_{2,j} \oplus (02 \cdot S_{3,j})$$

We know that 03 = 02 xor 01 then we can rewritten the above equation as:

$$S'_{0,j} = 02 \cdot (S_{0,j} \oplus S_{1,j}) \oplus S_{1,j} \oplus S_{2,j} \oplus S_{3,j}$$
$$S'_{1,j} = S_{0,j} \oplus 02 \cdot (S_{1,j} \oplus S_{2,j}) \oplus S_{2,j} \oplus S_{3,j}$$
$$S'_{2,j} = S_{0,j} \oplus S_{1,j} \oplus 02 \cdot (S_{2,j} \oplus S_{3,j}) \oplus S_{3,j}$$
$$S'_{3,j} = S_{0,j} \oplus S_{1,j} \oplus S_{2,j} \oplus 02 \cdot (S_{3,j} \cdot S_{0,j})$$

With the help of above equation we can easily get the values of mix column.

Add Round Key: In this transformation the 128 bit of stage are bitwise xor with the 128 bit of the round key. The operation is viewed as a column wise operation between the 4 bytes o a state column and one word of the round key.

3. PROPOSED AES PIPELINE STAGE

I this section we know the two general design for AES algorithm against our design. In the first design there is no pipeline and used iterative looping approach, while in the second design there is the pipeline buffer between ten AES design but in our design we use the pipeline architecture of 10 AES pipeline stage as well as in divides the AES into ten stages yielding an overall of 110 pipeline stages. As a result if we will see the speed in terms of throughput will increase.

In this section the architecture of proposed design is achieved where the process of AES is divided into 10 pipeline stages with equal amount of delay.

In addition we are using 10 AES block and these blocks are separated by pipeline stages.

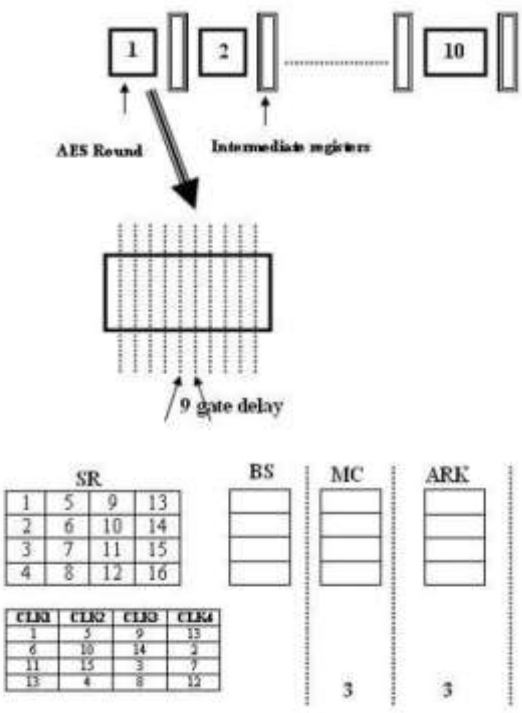


Figure 6 :Basic Rounds

Total amount of time required to encrypt N number of block can be evaluated. All block will take equal amount of time to encrypt the data thus one block will take one fourth part of total time including intermediate stage, Thus in the first round it will take maximum time for first byte to encrypt and after that the time will reduce because of pipeline stage.

4. CONCLUSION

The new hardware design architecture is proposed for advance encryption standard algorithm over GF (2^8) and compared to two AES hardware structure. First is the iterative looping and the second one is ten round pipeline approach.

The hardware implementation of the structure was conduct through SPARTEN 3E FPGA device. This design has the advantage to save the area increases the throughput and little bit decreases the latency. For saving the area we reordering the Byte Sub and Shift Row. The whole design is divided into 4 sub block (shift row, sub byte, mix column, add round key) and creating the deep pipeline structure for completing ten AES round.

REFERENCES

[1] Dr R.V. kshirsagar, M.V.Vyawahare "FPGA Implimentation of High Speed VLSI Architecture For AES Algorithm "2012 IEEE.

[2] Hassen Mestiri, Noura Benhadjyoussef, Mohsen Machhout and Rached Tourki "An FPGA Implimentation of the AES with fault detection Countermeasuer"2013 IEEE.

[3] Mr. Atul N Borkar "FPGA implementation of AES Algorithm" 2011 IEEE.

[4] Kaijie Wu, Ramesh Karri, Grigori Kuznetsov, Michael Goessel "Low Cost Concurrent Error Detection for the Advanced Encryption Standard" 2004 IEEE.

[5] Shaaban Sahmoud, Wisam Elmasry, Shadi Abudalfa "Advancement the security of AES against modern attacks by using variable key block cipher " IAJET vol. 3 no 1 march 2012.

[6] Alan Kaminsky, Micheal Kurdziel, Stanislaw Radziszowski "An overview of cryptanalysis research for the AES" IEEE 2010.

[7] Ashwini M. Deshpande, Mangesh S. Deshpande and Devendra N. Kayatanavar "FPGA Implementation of AES Encryption and Decryption " IEEE June 2009.

[8] *Advanced Encryption Standard (AES)*, Nov. 26, 2001.

[9] Kris Gaj, and Pawel Chodowiec "Hardware performance of the AES finalist's survey and analysis of results", George Mason University. Proc. 3rd Advanced Encryption Standard (AES) Candidate

Conference, New York, 2000:1-5.

[10] A. Elbirt “Recon_gurable Computing for Symmetric-Key Al-gorithms”, Ph.D. thesis, Department of Electrical Engineering, Worcester Polytechnic Institute, 2002.

[11] Elena Trichina, and Tymur Kokishko “Secure AES Hardware Module for Resource Constrained Devices”, Security in Ad-hoc and Sensor Networks. First European Workshop. ESAS 2004, Heidelberg, Germany, August 6, 2004; (3313):215- 229.

[12] Sumio Morioka, and Akashi Satoh “An Optimized S-BOX Circuit Architecture for low power AES Design”, IBM Research, Tokyo Research laboratories, CHES 2002; (2523):172-186.

[13] N. Sklavos, O. Koufopavlou, "Architectures and VLSI Implementations of the AES-Proposal Rijndael", IEEE Transactions on Computers, Vol. 51, Issue 12, pp. 1454-1459, 2002.

[14] Akashi Satoh, and Sumio Morioka “Unified Hardware Architecture for 128-Bits Block Ciphers AES and Camellia”, IBM Research, Tokyo Research laboratories, CHES 2003; (2779):304-318.