International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 7, July 2015, PP 140-146
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Efficient Area and High Speed Advanced Encryption Standard Algorithm

[1]G.Anjali, [2]Sudhir Dakey

[1]*Dept.of Electronics and communication engg, M.V.S.R Engineering college, Hyderabad*
[2]*Assistant Professor, Dept.of Electronics and Communication engg, M.V.S.R Engineering College, Hyderabad*

ABSTRACT

An efficient implementation of the Advanced Encryption Standard (AES) Algorithm. The presented architecture is adapted for AES encryption, encryption/decryption designs. The Sub, Inv Sub Bytes 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. We perform the same using ancient Vedic Mathematics techniques. The cryptographic unit involving mix columns & inverse mix columns for AES was designed.

Keywords: cryptography, vedic mathematics, composite field arithmetic.

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.

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, ShiftRows, MixColumns, AddRoundKey or their corresponding inverses during decryption.

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.

Address for correspondence:

golianjali@gmail.com

Subbytes/Inversebytes Transformation Using CFA

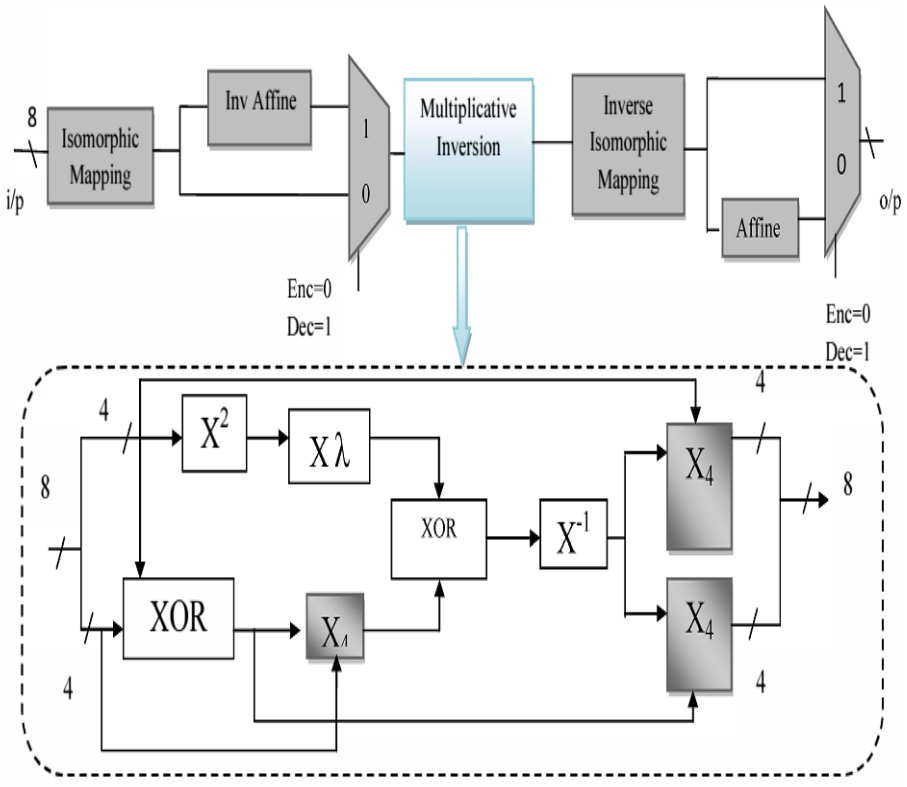

Fig1. *subbytes and invsubbyte block diagram*

S-box has the main two transfonnations. One is the multiplicative inversion and another one is the affine transfonnation. Fig shows subbyte and invsubbyte In Subbytes the operation is multiplicativeinversion to affine transformation [3]. In inv subbytes, the operation is inv affine transfonnation to m ultiplicative 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].

Shiftrow/Inverse Shiftrow Operation

Shiftrow 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 shiftrow 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.

Mix/Inverse Mixcolumns Operation

Mixcoloums

Mixcolumns (MC) and inverse mixcolumns (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 mixcolumns

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

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

Key Expansion Architecture

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 = RotWord(SubWord(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 SubWord function in (1) performs the SubBytes 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 SubWord function followed by a successive *XOR*ing to calculate each column wr,i from the previous column wr,i-1. The proposed KeyExpansion architecture computes the successive XORing in parallel with the SubWord function as shown in Fig. 2. Consequently, the remaining part is only to *XOR* the output of SubWord function with all four columns in parallel. Therefore, the critical path delay is decreased by 3 *XOR* gates through the parallelization of the KeyExpansion steps compared to the conventional KeyExpansion structure in [7].

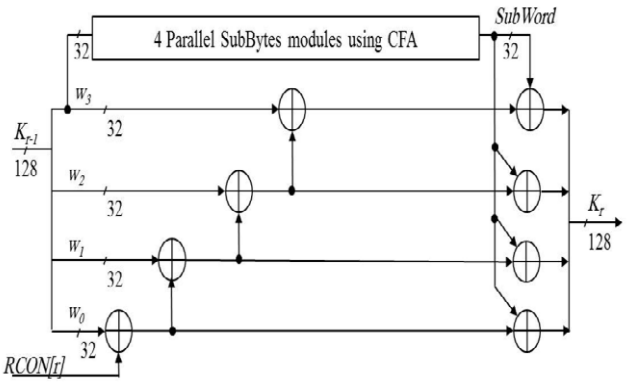

Fig2. *Key Expansion Architecture*

NOVEL METHOD FOR MC&IMC CALCULATIONS USING VEDIC MATHEMATICS

One of the crucial mathematical operation performed during the mix column step in AES, is the Galois field multiplication. Multiplication, being a tedious and a power hungry operation causes the

computation of mix columns and its inverse to be an even more arduous task. This is due to the fact that, it involves matrix multiplication. Therefore, there arises a necessity to ease the entire process and also overcome a few of the shortcomings of the previous methods mentioned in the previous section. In order to achieve the same, the Urdhwa Tiryakbhyam Sutra of Vedic Mathematics, is incorporated in our proposed architecture for mix columns and its inverse due to its excellence in terms of speed and area. The Urdhwa Tiryakbhyam Sutra is one of the significant sutras in ancient Vedic Mathematics. By its definition, Urdhwa Tiryakbhyam means "vertically crosswise". This implies that multiplication occurs between extreme bits of the multiplier and multiplicand. The major advantage of this algorithm is the availability of the product of two numbers in a single step. Also, since multiplication of two single bits reduces to a single AND operation, for a VLSI implementation this approach proves to be both area and speed efficient[8].

Let us assume M and N are the two numbers to be multiplied and are eight bits each. Let us assume P is the product achieved upon multiplication of M & N. The subscripts to both the letters M & N indicate the bit positions in both the numbers. The subscript '0' indicates the Least significant Bit (LSB) and the highest subscript indicates the Most Significant Bit (MSB). the multiplication of M and N, with dots in the first row representing the bits of M and dots in the second row representing the bits of N. Urdhwa Tiryakbhyam Sutra essentially comprises of logically cross-ANDing the bits of the multiplier and the multiplicand. These cross-ANDed results are termed as partial products. These partial products are immediately added so as to calculate each bit of the product. It must also be noted, that the addition of these partial products leads to the generation of carry's, denoted by Ci where i=0, 1, 2 3.

$$P0=M0*N0 \quad (1)$$

$$C1P1=(M1*N0)+(M0*N1) \quad (2)$$

$$C3C2P2=(M2*N0)+(M0*N2)+(M1*N1)+C1 \quad (3)$$

$$C5C4P3=(M3*N0)+(M2*N1)+(M1*N2)+(M0*N3)+C2 \quad (4)$$

$$C7C6P4=(M4*N0)+(M3*N1)+(M2*N2)+(M1*N3)+(M0*N4)+C3+C4 \quad (5)$$

$$C10C9C8P5=(M5*N0)+(M4*N1)+(M3*N2)+(M2*N3)+(M1*N4)+(M0*N5)+C5+C6 \quad (6)$$

$$C13C12C11P6=(M6*N0)+(M5*N1)+(M4*N2)+(M3*N3)+(M2*N4)+(M1*N5)+(M0*N6)+C7+C \quad (7)$$

$$C16C15C14P7=(M7*N0)+(M6*N1)+(M5*N2)+(M4*N3)+(M3*N4)+(M2*N5)+(M1*N6)+(M0*N7)+C9+C11 \quad (8)$$

$$C19C18C17P8=(M7*N1)+(M6*N2)+(M5*N3)+(M4*N4)+(M3*N5)+(M2*N6)+(M1*N7)+(M0*N7)+C10+C12+C14 \quad (9)$$

$$C22C21C20P9=(M7*N2)+(M6*N3)+(M5*N4)+(M4*N5)+(M3*N6)+(M2*N7)+C13+C15+C17 \quad (10)$$

$$C25C24C23P10=(M7*N3)+(M6*N4)+(M5*N5)+(M4*N6)+(M3*N7)+C16+C18+C20 \quad (11)$$

$$C27C26P11=(M7*N4)+(M6*N5)+(M5*N6)+(M4*N7)+C19+C21+C23 \quad (12)$$

$$C29C28P12=(M7*N5)+(M6*N6)+(M5*N7)+C22+C24+C26 \quad (13)$$

$$C30P13=(M7*N6)+(M6*N7)+C25+C27+C28 \quad (14)$$

$$P14==(M7*N7)+C29+C30 \quad (15)$$

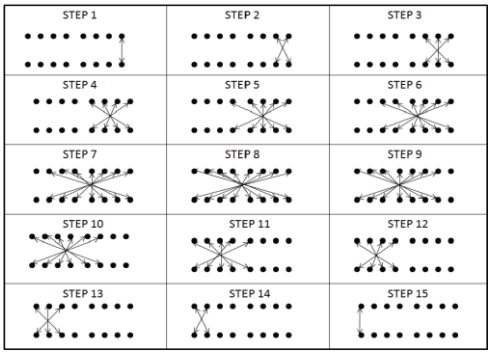

Fig3. *Depiction of Urdhwa Tiryakbhyam Sutra for multiplication of 8 bit numbers with the aid of dot diagrams.*

As mentioned above, the Urdhwa Tiryakbhyam sutra is a very efficient algorithm for multiplication and hence can be used in the Mix columns/Inverse Mix columns architecture as well. However, since we are required to perform Galois Field multiplication and not the regular multiplication, our proposed architecture slightly deviates from the Urdhwa Tiryakbhyam architecture. In our implementation, instead of adding the partial products which were computed in the intermediate stages, we logically XOR the same.

P0=M0*N0 (16)

P1=(M1*N0)+(M0*N1) (17)

P2=(M2*N0)+(M0*N2)+(M1*N1) (18)

P3=(M3*N0)+(M2*N1)+(M1*N2)+(M0*N3) (19)

P4=(M4*N0)+(M3*N1)+(M2*N2)+(M1*N3)+(M0*N4) (20)

P5=(M5*N0)+(M4*N1)+(M3*N2)+(M2*N3)+(M1*N4)+(M0*N5) (21)

P6=(M6*N0)+(M5*N1)+(M4*N2)+(M3*N3)+(M2*N4)+(M1*N5)+(M0*N6) (22)

P7=(M7*N0)+(M6*N1)+(M5*N2)+(M4*N3)+(M3*N4)+(M2*N5)+(M1*N6)+(M0*N7) (23)

P8=(M7*N1)+(M6*N2)+(M5*N3)+(M4*N4)+(M3*N5)+(M2*N6)+(M1*N7)+(M0*N7) (24)

P9=(M7*N2)+(M6*N3)+(M5*N4)+(M4*N5)+(M3*N6)+(M2*N7) (25)

P10=(M7*N3)+(M6*N4)+(M5*N5)+(M4*N6)+(M3*N7) (26)

P11=(M7*N4)+(M6*N5)+(M5*N6)+(M4*N7) (27)

P12=(M7*N5)+(M6*N6)+(M5*N7) (28)

P13=(M7*N6)+(M6*N7) (29)

P14==(M7*N7)+C29+C30 (30)

Encryption Round Architecture:

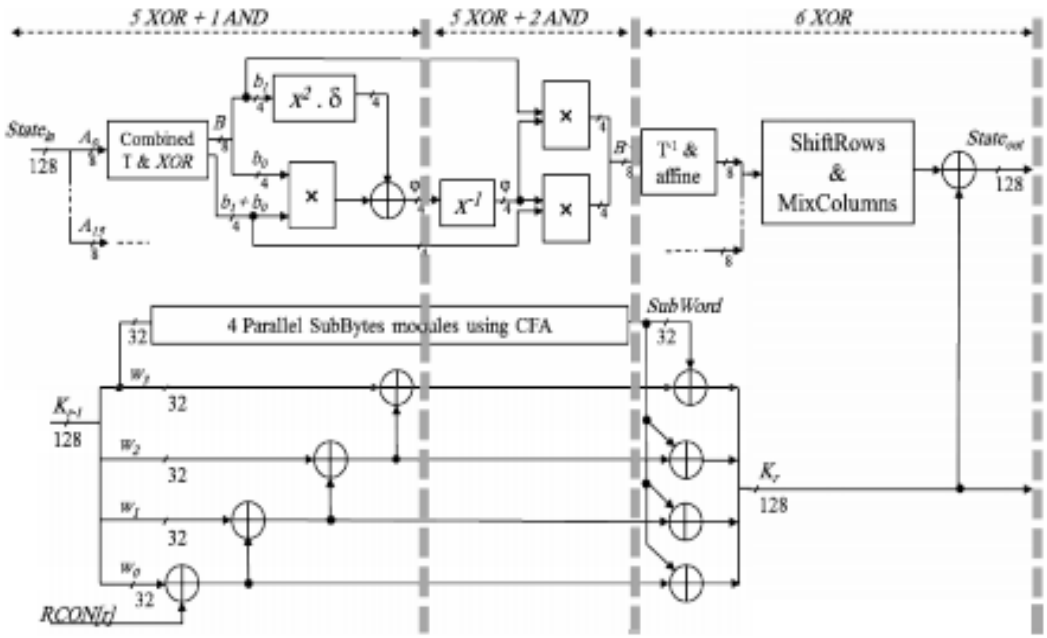

Fig4. *AES Encryption Round Architecture*

AES round architecture adopts the subpipelining 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:

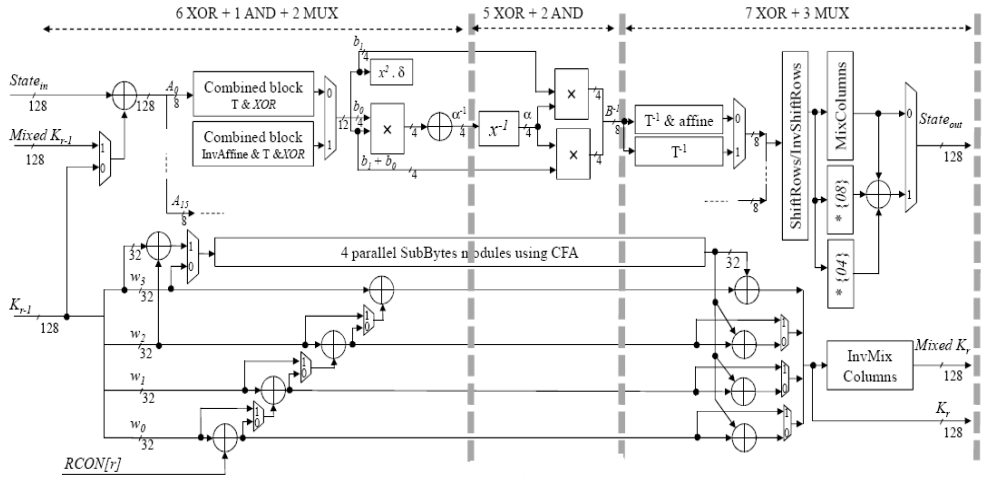

Fig5. *AES encryption /decryption round architecture*

AES decryption process performs the inverse encryption transformations in the following reverse order: AddRoundKey, InMixColumns, InvShiftRows, InvSubBytes. Furthermore, the parallel KeyExpansion 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 [1].

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.

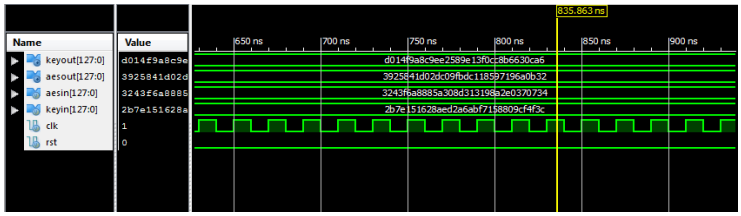

Fig6. *Simulation of AES Encryption Algorithm*

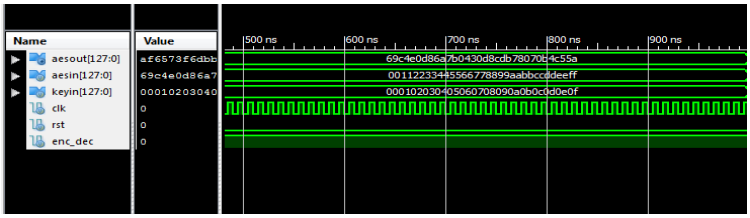

Fig7. *Simulation of AES Encryption/Decryption Algorithm*

Table1. *Comparison between Existing and Modified Versions*

Results comparison of encryption design

parameters	Existing method	Modified method
Slices	40691	9000
Delay(ns)	16.425	7.17
Throughput(mbps)	410.16	939.45
Efficiency(mbps/slice)	0.01	0.10

Results comparison of encryption/decryption design

parameters	Existing method	Modified method
Slices	42017	4733
Delay(ns)	9.868	4.457
Throughput(mbps)	180.14	398.82
Efficiency(mbps/slices)	0.0042	0.08

CONCLUSION

In this we have proposed a novel and efficient architecture using Vedic mathematics for performing mix and inverse mix column computations. AES encrypt round and adapted for integrated AES encrypt/decrypt. The Sub Bytes/InvSubBytes 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] Kit Choy Xintong "Understanding Aes Mix-Columns Transformation Calculation ",University Of Wollongong.

[6] Kit Choy Xintong "Understanding Aes Inverse Mix-Columns Transformation Calculation ", University Of Wollongong.

[7] 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.

[8] Sushma R Huddar, Sudhir Rao Rupanagudi, Ramya Ravi, Shikha Yadav & Sanjay Jain"Novel Architecture for Inverse Mix Columns for AES using Ancient Vedic Mathematics on FPGA."2013 IEEE.

[9] Salma Hesham, Mohamed A. Abd El Ghany." High Throughput Architecture for the Advanced Encryption Standard Algorithm"2014 IEEE.