ISSN: 2456-043X

IJESAR

International Journal of Engineering Science and Generic Research (IJESAR)
Available Online at www.ijesar.in
Volume 2; Issue 4; July-August; 2016; Page No. 27-32

# Design and Implementation of Area Efficient S-Box for Fully Pipelined AES

[1]Mrs.Kondisetti Jyothi, [2]Mr.Pattem Sampath Kumar, [3]Mrs.Velaga Rajya Lakshmi,

[1]Student, Dept. of ECE, Malla Reddy Institute of technology, Hyderabad, India, svvnjyothi@gmail.com

[2]Associate Professor & HOD, Dept. of ECE, Malla Reddy Institute of technology, Hyderabad, India,Email:sampath440@yahoo.co.in

[3]Assistant Professor, Dept. of ECE, Malla Reddy Institute of technology, Hyderabad, India,Email:rajimp.velaga@gmail.com

## Abstract

Encryption will convert the data in such a manner that only a person who has special knowledge of reading it can be able to read it. The Advanced Encryption Standard (AES) is considered to be the strongest encryption technique in cryptography. Advanced Encryption Standard (AES) is a symmetric key block cipher which will encrypt as well as decrypt the data block. The pre-computed values stored in a ROM based lookup table. In this implementation, all 256 values are stored in a ROM such implementation is expensive in terms of hardware.

A more refined way of implementing the S-Box by using Composite field method through combinational logic. This S Box has the advantage of having small area occupancy. Hence it delivers high throughput optimizes the delay and reduces the area. The 128-bit Advanced Encryption Standard (AES) of fully Pipelined AES with Compact S-Box is simulated and synthesized by Xilinx 13.2 tool.

**Keywords:** VLSI, AES, ENCRYPTION, DECRYPTION.

## Introduction

Cryptography information need is one to be of the secured security from mechanism unauthorized to protect party. The data from public access. Cryptography is a Greek origin word which means" secret-writing" to make the data secure and immune to attacks. In past days cryptography was used for the top secret communication between the two people. But now day's cryptography has changed into algorithm depending upon the demand of the people. The algorithm has two process namely encryption and decryption. The first process is encryption which will convert the original plain text into cipher text which is secured text The second process is decryption which will recover the original plain text by converting the cipher text into original plain text. There are mainly two types of encryption algorithms: symmetric key algorithm and asymmetric key algorithm. Symmetric key algorithm is also called private key algorithm which involve only one key both for the encryption as well as for decryption. Asymmetric key algorithm is also called public key algorithm which involves two keys one for the encryption and another for the decryption. The private key algorithm is less complex and easy to implement as compared to public key algorithm. Also private key algorithms are fast as compared to public key algorithm. There are different types of symmetric key algorithms such as: DES, Triple-DES, AES, RC4 etc and asymmetric key algorithms such as: RSA, Elliptic Curve Cryptography etc.

## DESCRIPTION OF AES ALGORITHM

The Advanced Encryption Standard (AES) is a symmetric key block cipher algorithm developed by Joan Daemon of Proton World International and Vincent Fijmen of Katholieke University at Leuven. Advanced Encryption Standard (AES) is block cipher, which means it works on the fixed length group of bits, which are called blocks. AES algorithm performs both the encryption and decryption of the data. Encryption will convert the data into a unintelligible form called cipher text. Decryption converts the cipher-text into its original form called plain-text. AES supports block data of 128-bits, 192-bits or 256-bits with the private key of length 128-bits, 192-bits or 256-bits. Each iteration in an algorithm is called as round and it has l0, 12 or 14 rounds for processing data blocks of 128-bits, 192-bits or 256- bits respectively. In this paper, we have used key length of 128 bits (AES-128) that is we will be working with block data of 128-bits. As we know 128-bits is equivalent to 16 bytes that are arranged in matrix form know as state in the algorithm. The matrix size depends upon the block size being used, composed of 4 rows and Nb columns. Nb mainly represents the number of bits in the data block, divided by 32 since 4 bytes represent the 32-bits. As we are working with 128-bits data block the matrix of the state consists of 4 rows and 4 columns. The key is also grouped in the same fashion as that of data block, whereas Nk represents number of columns. Nr represents the number of rounds which will be there during the encryption and decryption process in AES algorithm

On the encryption side there will be 4 transformations: AddRoundkey, Sub Bytes, Shift Row and Mix Columns transformation. In the last round Mix Columns transformation is suppressed. The decryption side will uses the inverse transformations: InvAddRoundkey, InvSubBytes, invshiftRow and InvMixColumns transformation. As it was in encryption side, the Inv Mix Columns transformation is also suppressed in decryption side. Basic structure of AES algorithm

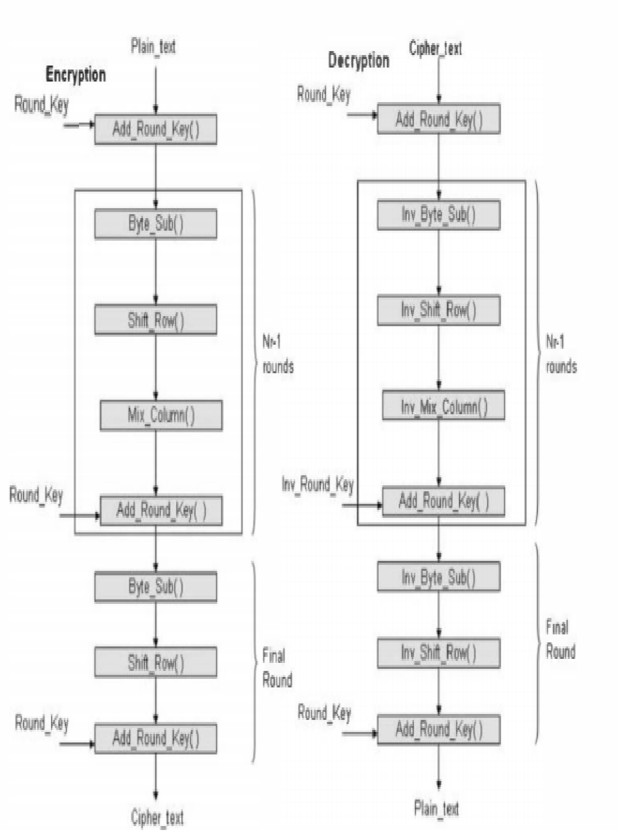


**Figure 1: AES algorithm**

### A. AES encryption:

At encryption side AES performs the four transformations which are as follows:

### 1. Sub Bytes Transformation:

The Sub Bytes transformation is a non-linear byte substitution that operates independently on each byte of state matrix using a substitution table called as S-Box. Each state byte is replaced by another byte with the help of S-box[4]. The substitution follows a matrix, where the first hexadecimal value corresponds to row positioning and second hexadecimal value corresponds to column positioning.

### 2. Shift Rows Transformation:

In shift Rows transformation the bytes of the state are shifted cyclically to the left by a fixed number called offset [3].Row 0 is not shifted. Row 1 is shifted by 1 byte. Row 2 is shifted by 2 bytes. Row 3 is shifted by 3 bytes.

### 3. Mix Columns Transformation:

The Mixcolums transformation is based on Galois Field Multiplication. In this transformation each column of the state is considered as four-term polynomial over GF($2^{8)}$ and multiplied modulo $X^4$ + 1 with a fixed polynomial a(x), a(x) is given as, a(x) ={03} x3+{0l} x2+{0l} x+{02}.

### 4. AddRoundKey Transformation:

AddRoundKey transformation is an XOR operation between the state and round key. The round key for each round is generated by the key expansion algorithm. The total eleven 128-bits round keys are required for the encryption process. The matrix of the keys is represented by w columns or kx, y cells. The XOR operation is performed on byte where S'x, y new byte is given by Sx, y EB kx, y.

### B. AES decryption:

At decryption side AES performs the four inverse transformations which are as follows:

### 1. InvSubBytes Transformation:

The InvSubBytes transformation is also a non-linear byte substitution that operates independently on each byte of state matrix using a inverse substitution table called as InvSBox [4].

### 2. InvShiftRows Transformation:

InvShiftRows transformation is an inverse of Shift Row transformation. Row 0 is not shifted. Row 1 is shifted right by 1byte. Row 2 is shifted right by 2 bytes. Row 3 is shifted right by 3 bytes.

### 3. InvMixColumns Transformation:

The InvMixcolumns transformation multiplies the polynomial formed by each column of the State with a·l(x) modulo x4+1, where a-l(x) = {Ob}x3 + {Od}x2 + {09}x + {Oe}.

### 4. InvAddRoundKey Transformation:

AddRoundkey transformation is its known inverse function because the XOR operation is its own reverse. The round keys have to be selected in reverse order.

## III. AES IMPLEMENTATION USING FULLY PIPELINED ARCHITECTURE

The goal of AES implementation using fully pipelined architecture is to achieve the highest throughput and to optimize the delay. The block diagram of AES implementation using fully pipelined architecture basically two types of pipelining are possible in AES implementation: inner round pipelining and outer round pipelining. Here in our design we have used the outer round pipelining. Here pipelining is done after each round hence it is called outer round pipelining and also pipelining is done in non-feedback mode so it is called fully pipelined AES. In fully pipelined AES pipeline registers are placed after each round. The pipelining between each round will achieve high performance implementation both at encryption and decryption side. Although implementing an iterative pipelining based approach is one option, for simplicity and clarity, we have used the fully expanded implementation for all ten rounds. The data generated in each individual round is utilized as an input data to next round. Here placing the pipeline registers is the key of achieving better performance. Pipeline registers are basically used for intermediate data processing.

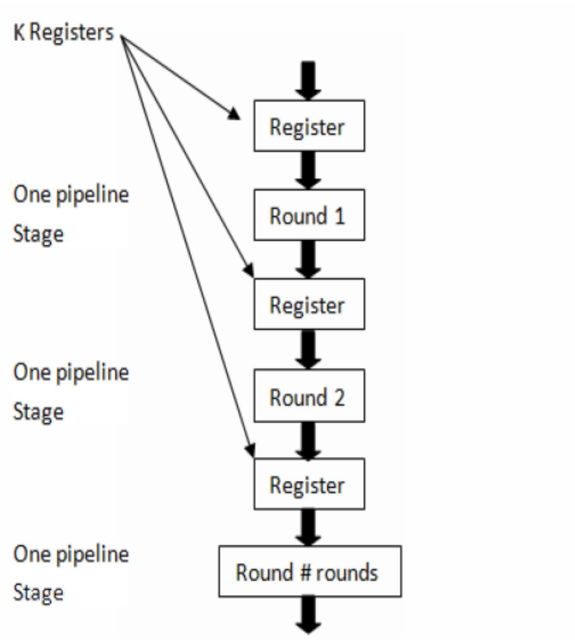
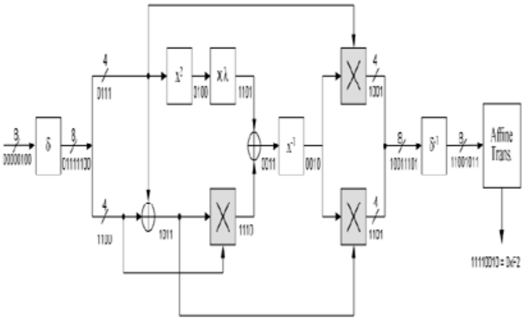


**Figure 2: pipelined AES**

## IV. COMPOSITE FILED S BOX

One of the most common and straight forward implementation of the S-Box for the Sub Byte operation which was done in previous work was to have the pre-computed values stored in a ROM based lookup table. In this implementation, all 256 values are stored in a ROM and the input byte would be wired to the ROM's address bus. However, this method suffers from an unbreakable delay since ROMs have a fixed access time for its read and write operation. [3] Furthermore, such implementation is expensive in terms of hardware. A more refined way of implementing the S-Box is to use combinational logic. This S Box has the advantage of having small area occupancy



### Isomorphic Mapping and Inverse Isomorphic Mapping

Computation of the multiplicative inverse in composite fields cannot be directly applied to an element which is based on GF($2^8$). That element has to be mapped to its composite field representation via an isomorphic function, δ. Likewise, after performing the multiplicative inversion, the result will also have to be mapped back from its composite field representation to its equivalent in GF($2^8$) via the inverse isomorphic function, δ-1. Both δ and δ-1 can be represented as an 8x8 matrix. Let *q* be the element in GF($2^8$), then the isomorphic mappings and its inverse can be written as δ**q* and δ-1**q*, which is a case of matrix multiplication as shown below, where *q*7 is the most significant bit and *q0* is the least significant bit

$$\delta \times q = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} \qquad \delta^{-1} \times q = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix}$$

The matrix multiplication can be translated to logical XOR operation. The logical form of the matrices above is shown below.

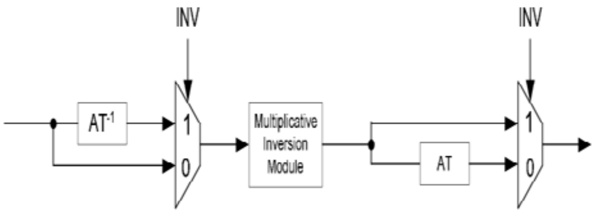
$$\delta \times q = \begin{pmatrix} q_7 \oplus q_5 \\ q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_6 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_6 \oplus q_4 \oplus q_1 \\ q_6 \oplus q_1 \oplus q_0 \end{pmatrix} \qquad \delta^{-1} \times q = \begin{pmatrix} q_7 \oplus q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_2 \\ q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_0 \end{pmatrix}$$

### B. Affine Transformation and inverse Affine Transformation

$$AT(a) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

$$AT^{-1}(a) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

The AT and AT-1 are the Affine Transformation and its inverse while the vector *a* is the multiplicative inverse of the input byte from the state array. From here, it is observed that both the Sub Byte and the InvSubByte transformation involve a multiplicative inversion operation. Thus, both transformations may actually share the same multiplicative inversion module in a combined architecture. An example of such hardware architecture is shown below. Switching between SubByte and InvSubByte is just a matter of changing the value of INV. INV is set to 0 for Sub Byte while 1 is set when InvSubByte operation is desired.



### C. Addition

Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation between the 2 elements.

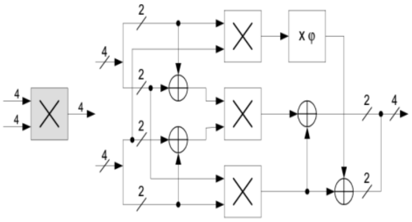
### D. Squaring

$$k_3 = q_3$$
$$k_2 = q_3 \oplus q_2$$
$$k_1 = q_2 \oplus q_1$$
$$k_0 = q_3 \oplus q_1 \oplus q_0$$

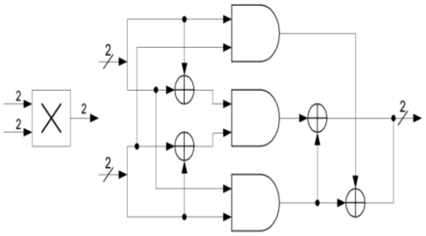
### E. Multiplication with constant, λ

$$k_3 = q_2 \oplus q_0$$
$$k_2 = q_3 \oplus q_2 \oplus q_1 \oplus q_0$$
$$k_1 = q_3$$
$$k_0 = q_2$$

### F. GF (2^4) Multiplication



### G. GF (2^2) Multiplication



### H. Multiplication with constant ϕ

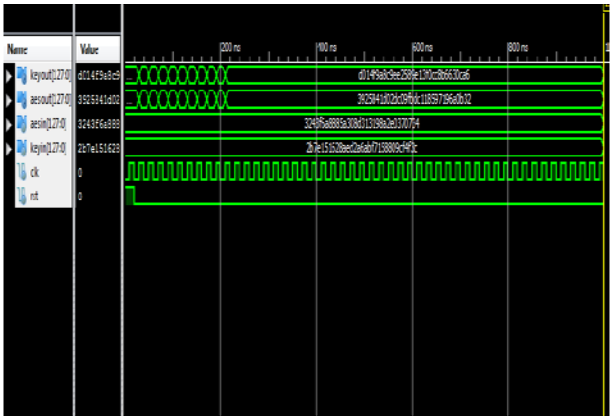
$$k_1 = q_1 \oplus q_0$$
$$k_0 = q_1$$

### I. Multiplicative Inversion in GF (2^4)

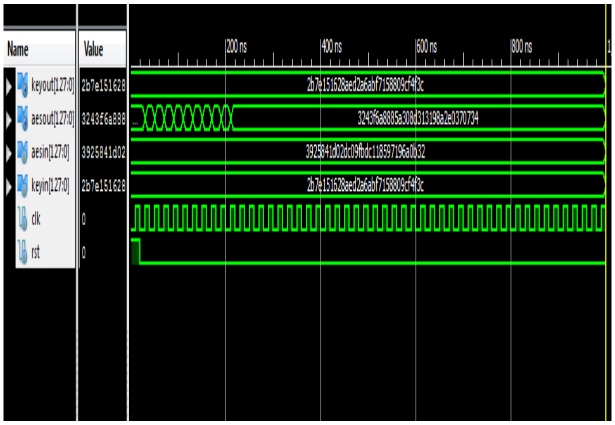
$$q_3^{-1} = q_3 \oplus q_3 q_2 q_1 \oplus q_3 q_0 \oplus q_2$$
$$q_2^{-1} = q_3 q_2 q_1 \oplus q_3 q_2 q_0 \oplus q_3 q_0 \oplus q_2 \oplus q_2 q_1$$
$$q_1^{-1} = q_3 \oplus q_3 q_2 q_1 \oplus q_3 q_1 q_0 \oplus q_2 \oplus q_2 q_0 \oplus q_1$$
$$q_0^{-1} = q_3 q_2 q_1 \oplus q_3 q_2 q_0 \oplus q_3 q_1 \oplus q_3 q_1 q_0 \oplus q_3 q_0 \oplus q_2 \oplus q_2 q_1 \oplus q_2 q_1 q_0 \oplus q_1 \oplus q_0$$

## V. SIMULATION RESULTS

### Encryption



Decryption



## VI. SYNTHESIZED RESULTS

| | No of Slices | No of 4 input LUT'S |
|---|---|---|
| AES Encryption(S-box look up table) | 16292 | 30415 |
| AES Decryption(S-box look up table) | 19014 | 36778 |
| AES Encryption(Compact S-Box) | 10260 | 19764 |
| AES Decryption(Compact S-Box) | 13730 | 25502 |

## VII. CONCLUSION

In this paper we have shown the simulation results of AES encryption and AES decryption. We will be doing the full outer round pipelining for all the 10 rounds in order to achieve optimize the delay. The pre-computed values stored in a ROM based lookup table. In this implementation, all 256 values are stored in a ROM such implementation is expensive in terms of hardware. A more refined way of implementing the S-Box is to use composite field S-Box through combinational logic. This S Box has the advantage of having small area occupancy which saves the area in terms number of logic slice used and number of four input look up table by keeping optimizes delay.

## VII. REFRENCES

1. Manjesh. K. N and R K Karunavathi ," Secured High throughput implementation of AES Algorithm ",Internat- onal Journal of Advanced Research in Computer Science and Software Engineering, vol.5 pp. 1193-1198,May 2013 .
2. Hoang Trang and Nguyen Van Loi, "An efficient FPGA implementation of the Advanced Encryption Standard (AES) algorithm", IEEE Transactions, vol. 20 pp. 978-1-4673-0309-5, 2012.
3. Shylashree .N, Nagarjun Bhat and V. Shridhar, " FPGA IMPLEMENTATIONS OF ADVANCED ENCRYPTION STANDARD", International Journal of Advances in Engineering & Technology,Vol. 3, Issue 2,pp. 265-2 ,May 2012.

4. P. V. Sriniwas Shastry and M .S. Sutaone, "A Sub-pipelined Implementation of AES for All Key Sizes", Intl. Conf. on Advances in Computer, Electronics and Electrical Engineering, ISBN: 978-981-07-1847-3, 2012.
5. Otavio S. M. Gomes, Robson L. Moreno and Tales C. Pimenta, "A Fast Cryptography Pipelined Hardware developed in FPGA with VHDL", Universidade Federal de Jtajuba - UNIFEI,2011.
6. AJ Elbert, W Yip, B Chetwynd and C Paar, " An FPGA Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists",IEEE Transactions on VLSI,2010.
7. DAEMEN, J. AND RUMEN, V. A Specification for the AES Algorithm. NIST (National Institute of Standards and Technology), 2010.
8. Paolo Maistri and Re' gis Leveugle, Member, IEEE "Double-Data Rate Computation as a Countermeasure against Fault Analysis" IEEE TRANSACTIONS ON COMPUTERS, VOL. 57, NO. II, NOVEMBER 2008.
9. S.M. Yoo, D. Kotturi, D.W. Pan and J. Blizzard,"An AES crypto chip using a high-speed parallel pipelined architecture", Microprocessors and Microsystems 29 (2005) 317-326, 2005.
10. Kimmo Jarvinen, Matti Tamika and Jorma Skytta, "Comparative Survey of High Performance Cryptographic Algorithm Implementations on FPGAs" , IEEE Proceedings - Information Security, vol. 152, no. I, Oct. 2005,pp. 3-12.
11. DAEMEN,J. AND RUMEN,V," The design of Rijndael: AES – The Advanced Encryption Standard",Springer-Verlag 2002..
12. FIPS-197, Federal Information Processing Standards Publication FIPS-I97, Advanced Encryption Standard (AES), 2001.

## Author Details:

| | |
|---|---|
| | **Mrs. K.V.V.N. Jyothi** is working towards Master of Technology in Embedded Systems & VLSI at prestigious Malla Reddy Institute of Technology, Hyderabad.<br>E-mail: svvnjyothi@gmail.com |
| | **Mrs. V.Rayja Lakshmi** is presently working as an Associate Professor at Malla Reddy Institute of Technology, Hyderabad in ECE Department. |
| | **Mr. P.Sampath Kumar** is presently working as an Associate Professor & HOD of Electronics Communication Engineering in prestigious Malla Reddy Institute of Technology, Hyderabad. |