ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 3, March 2017

Design of AES-256 with Three Stage Pipelining S-Box Using Logic Gates

[1]Pujari Munaswamy, [2]N.Dilipkumar, [3]N.Pushpalatha
[1] (M.Tech) [2][3]Assistant Professor
[1][2][3]Dept of ECE, Annamacharya Institute Of Technology & Sciences, Tirupati

Abstract: — **The cipher is the component which is responsible for performing encryption or decryption on blocks of input data, while the key expander is responsible for preparing the input key for use by the cipher in each round. The Advance Encryption Standard (AES) is composed of four different functions that are repeated in a number of rounds. These are byte substitution, shift row, mix column, and add round key. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. The number of rounds in AES is variable and depends on the length of the key, i.e10 rounds for 128-bit key, 12 rounds for 192-bit key and 14 rounds for 256-bit key. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. When a key of size 256 bits is used, the number of rounds are repeated is equal to 14. The function Sub Bytes is the only non-linear function in AES, operating on each of the state bytes independently. The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns. It substitutes all bytes of the State using a look-up table called S-Box. A more suitable method of implementing the S-Box is to use composite field approach for calculating multiplicative inverse in a Galois Field GF(28) followed by an affine transformation in the binary extension field , which is based on combinational logic. An efficient design is three stages pipelining for the byte substitution phase. The proposed Three Stages Pipelining S-Box Using Logic gates and efficient in terms of delay and area.**

Index Terms:— **S-box, encryption, decryption.**

I. INTRODUCTION

World wise the more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES. Because of DES data(64 bit) was needed as its key size(56bit) was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow[5].

In Network Security the Cryptography plays an important role in the security of data. Cryptography is derived from the Greek words: kryptos, "hidden", and graphein, "to write" or "hidden writing". It is useful to us for the purpose of store sensitive information or transmits it across insecure networks so that unauthorized persons cannot read it. Cryptography involves all legitimate users of information having the keys required to access that information. Based on keys types the encryption algorithm can be classified into two groups: symmetric encryption with private key algorithms (the sender and recipient must have the same key in order to encode or decode the protected information) and asymmetric encryption with public key algorithms(the sender and recipient have different keys respective to the communication roles they play).Symmetric key algorithms are in general much faster to execute electronically than asymmetric key algorithms[3].

In January 1997 NIST (National Institute of Standards and Technology) issued a public request for candidates to replace the aging DES, which resulted in 15 viable submissions from 12 countries. In October 2000 NIST announced the winner of the contest as Rijndael, a program created by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, had been accepted as the new standard, or the Advanced Encryption Standard (AES)[6]. the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128, 192, 256 bits,

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 3, March 2017

Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES-256 respectively. A number of AES parameters depend on the key length. If the key size used is 128 then the number of rounds is 10 where as it is 12 and 14 for 192 and 256 bits respectively[7].

II. THE AES-256 ALGORITHM

AES is based on a design principle known as a substitution-permutation network. The AES-256 algorithm is mainly consists of two major parts: Cipher and Key Expansion. The cipher is the component which is responsible for performing encryption or decryption on blocks of input data. In encryption process converts plaintext to cipher text using key while Decryption process converts cipher text to plaintext using key. Key Expansion generates a Key Schedule that is used in Cipher procedure [1].

A. AES-256 Encryption process:

The Encryption process consists of a plaintext is 128 bits and key is 256 bits and the number of rounds in AES 256 is 14. The AES-256 encryption process having four different functions are repeated in a number of rounds, these are sub byte operation, shift row operation, mix column and add round key operations. In the first round we are having all the five operations like Pre round operation, sub byte operation, shift rows operation, mix columns and Add round key operations. From 2nd round to 13th round have four operations sub byte operation, shift rows operation, mix columns and Add round key operations.The final 14th round consists only three operations. i.e sub byte operation, shift rows and Add round key operations respectively[4][6].

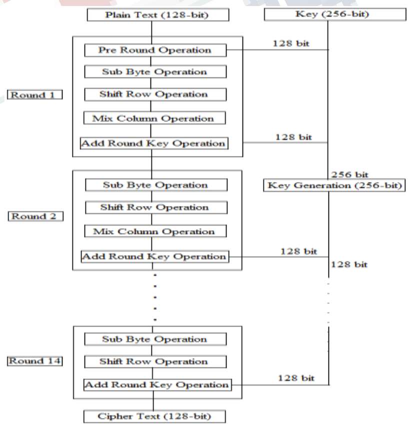
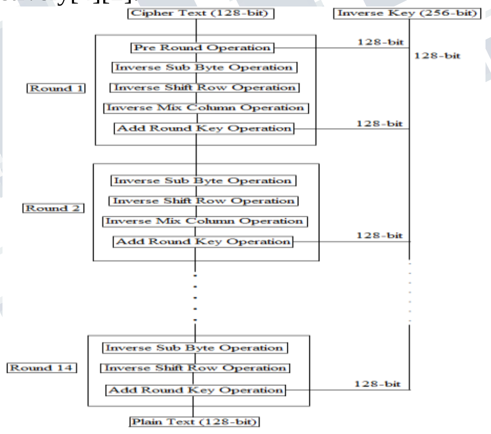


Fig.2.1 AES-256 Encryption process

B. AES-256 Decryption process:

In the Decryption process Cipher text is given as input. The input key is given to the key generation module to generate new keys are given as input to inverse mix columns, which gives key for the fourteen rounds of decryption. In AES-256 decryption process we are having four different functions are repeated in a number of rounds, these are inverse sub byte operation, inverse shift row operation, inverse mix column and inverse add round key operations. The first round consists of all the five operation like Pre round operation, inverse sub byte operation, inverse shift rows operation, inverse mix columns and inverse Add round key operations. From 2nd round to 13th round have four operations, like inverse sub byte operation, inverse shift rows operation, inverse mix columns and inverse Add round key operations. The last 14th round consists only three operations inverse sub byte operation, inverse shift rows and inverse Add round key operations respectively[4][2].


Cipher Text (128-bit)
Inverse Key (256-bit)
Pre Round Operation
128-bit
128-bit
Inverse Sub Byte Operation
Round 1
Inverse Shift Row Operation
Inverse Mix Column Operation
Add Round Key Operation
128-bit
Inverse Sub Byte Operation
Round 2
Inverse Shift Row Operation
Inverse Mix Column Operation
Add Round Key Operation
128-bit
Inverse Sub Byte Operation
Round 14
Inverse Shift Row Operation
Add Round Key Operation
128-bit
Plain Text (128-bit)
Fig.2.2 AES-256 Decryption process

C. Key expansion process:

In the Key expansion process, the input key of 256 bit is divided into eight parts. Each part consists of 32 bits, arranged in a matrix 4 X 8. Last column is taken as a row matrix 1 X 4 and performed shift row operation. The output of shift rows operation is given to S-Box to perform a sub byte operation. XOR with the round constant at MSB side 8 bits of sub byte operation output (i.e. round constant value changes for every round). The XOR round constant output is XOR with input key of 0th column and the output is taken as generated new key of 0th column, and the 0th column of new key is XOR with 1st column of input key and the output is taken as generated new key of 1st column of new key. Similarly 1st column of new key is XOR with 2st column of input key to give 2nd column of new key,2nd column of new key is XOR with 3rd column of input key gives 3rd column of new key. The generated new key of 3rd column is given to S box to perform XOR operation. The output of S box is XOR with 4th column of input key which gives 4th column of new key, 4th column of new key is XOR with 5th column of input key which gives 5th column of new key,5th column of new key is XOR with 6th column of input key which gives 6th column of new key and finally 6th column of new key is XOR with 7th column of input key which gives 7th column of new key. This process is shown in Fig 2.3 AES-256 Key expansion, this way we generate new keys of 256 bits in AES 256 algorithm by attaching the eight obtained columns of new key[1][7].

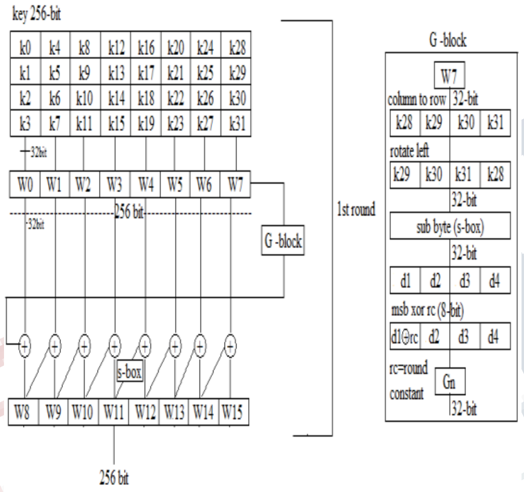


Fig.2.3 AES-256 Key expansion

III. S-BOX USING COMBINATIONAL LOGIC

The Example for S-Box using combinational logics computing the Sub Byte operation is show in Fig 3.1,the propagation of the input data of 0X4 into a composite field based 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. The input data will first undergo the multiplicative inversion. The values at which the high and low nibbles are transformed to are indicated by the 4 bit numbers outside of the logical blocks. The example can be worked by hand since the tables containing the results for GF(2^4) multiplication and multiplicative inverses are provided. After the inverse isomorphic mapping operation of the multiplicative inversion module, the Affine Transformation is applied to the multiplicative inverse to yield the S-Box substituted value for the given input of 0 X CB. Doing so yields an output of 0XF2 which agrees with the S-Box table provided. A more refined way of implementing the S-Box is to use combinational logic and this type of S-Box has the advantage of having small area occupancy [1][2].

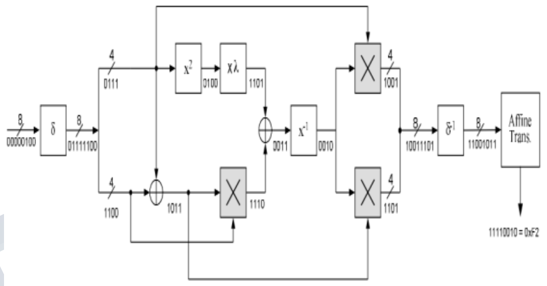


Fig 3.1 Example for S-Box using combinational logics computing the SubByte operation.

A. The Sub Byte and Inverse Sub Byte Transformation:

The Sub Byte transformation is calculate by taking the multiplicative inverse in GF(2^8) followed by an affine transformation. The Inv Sub Byte transformation is inverse process of Sub Byte transformation, the inverse affine transformation is applied first prior to claculating the multiplicative inverse. The steps involved for both transformation is shown below.

Sub Byte: Multiplicative Inversionin GF(2^8) → Affine Transformation

Inverse Sub Byte: Inverse Affine Transformation → Multiplicative Inversion in GF(2^8) The Affine transformation and inverse Affine transformation can be represented in matrix form and it is shown below.

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

The AT and AT^{-1} are the Affine Transformation and inverse Affine Transformation 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

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 3, March 2017

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 Fig 3.2. Switching between SubByte and InvSubByte is just changing the value of INV. INV is set to 0 for Sub Byte while 1 is set when Inv Sub Byte[2][4].

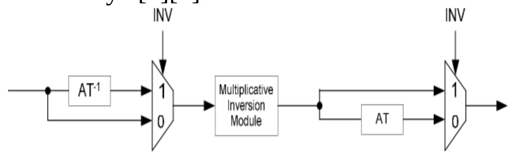


Fig.3.2 Combined Sub Byte and Inverse Sub Byte sharing a common multiplicative inversion module.

B. *Isomorphic Mapping and Inverse Isomorphic Mapping:*

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

$$\delta \times q = \begin{pmatrix} 1&0&1&0&0&0&0&0\\ 1&1&0&1&1&1&1&0\\ 1&0&1&0&1&1&0&0\\ 1&0&1&0&1&1&1&0\\ 1&1&0&0&0&1&1&0\\ 1&0&0&1&1&1&1&0\\ 0&1&0&1&0&0&1&0\\ 0&1&0&0&0&0&1&1 \end{pmatrix} \times \begin{pmatrix} q_7\\ q_6\\ q_5\\ q_4\\ q_3\\ q_2\\ q_1\\ q_0 \end{pmatrix} \quad \delta^{-1} \times q = \begin{pmatrix} 1&1&1&0&0&0&1&0\\ 0&1&0&0&0&1&0&0\\ 0&1&1&0&0&0&1&0\\ 0&1&1&1&0&1&1&0\\ 0&0&1&1&1&1&1&0\\ 1&0&0&1&1&1&1&0\\ 0&0&1&1&0&0&0&0\\ 0&1&1&1&0&1&0&1 \end{pmatrix} \times \begin{pmatrix} q_7\\ q_6\\ q_5\\ q_4\\ q_3\\ q_2\\ q_1\\ q_0 \end{pmatrix} \quad (2)$$

Then the above matrix becomes

$$\delta \times q = \begin{pmatrix} q_7 \oplus q_5\\ q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1\\ q_7 \oplus q_5 \oplus q_3 \oplus q_2\\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1\\ q_7 \oplus q_6 \oplus q_2 \oplus q_1\\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1\\ q_6 \oplus q_4 \oplus q_1\\ q_6 \oplus q_1 \oplus q_0 \end{pmatrix} \quad \delta^{-1} \times q = \begin{pmatrix} q_7 \oplus q_6 \oplus q_5 \oplus q_1\\ q_6 \oplus q_2\\ q_6 \oplus q_5 \oplus q_1\\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1\\ q_5 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1\\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1\\ q_5 \oplus q_4\\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_0 \end{pmatrix} \quad (3)$$

C. *Addition in GF(2^4):*

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

D. *Squaring in GF(2^4):*

Let k = q^2 where k and q is an element in GF(2^4), represented by the binary number of {k_3 k_2 k_1 k_0}$_2$ and {q_3 q_2 q_1 q_0}$_2$ respectively**.**

$k_3=q_3$.......................... (4)

$k_2=q_3 \oplus q_2$.................. (5)

$k_1=q_2 \oplus q_1$................. (6)

$k_0=q_3 \oplus q_1 \oplus q_0$......... (7)

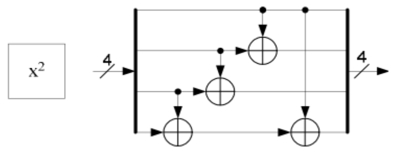


Fig.3.3 Hardware diagram for Squarer in GF(2^4).

E. *Multiplication with constant, λ :*

Let k = qλ, where k = {k3 k2 k1 k0}$_2$, q = {q3 q2 q1 q0}$_2$ and
λ = {1100}$_2$ are elements of GF(2^4).

$k_3=q_2 \oplus q_0$.................... (8)

$k_2=q_3 \oplus q_1 \oplus q_0$........... (9)

$k_1=q_3$............................. (10)

$k_0=q_2$………………... (11)

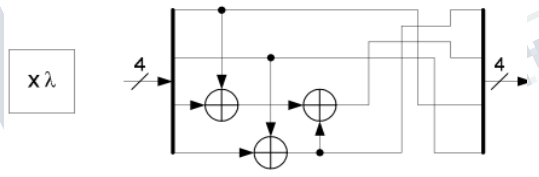


Fig.3.4. Hardware diagram for multiplication with constant λ

F. *GF(2^4) Multiplication:*

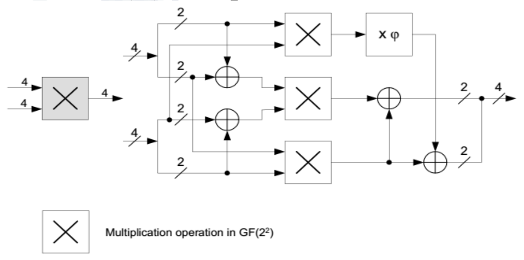


Fig.3.5. Hardware implementation of multiplication in GF(2^4)

G. *GF(2^2) Multiplication:*

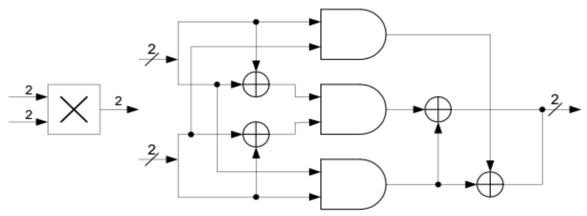


Fig.3.6. Hardware implementation of multiplication in GF(2)

H. Multiplication with constant φ:

The formula for computing multiplication with φ can be derived and is shown in equation (12) & (13).

$k_1=q_1 \oplus q_0$........................ (12)
$k_0=q_1$...................... (13)

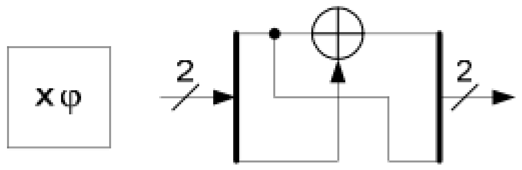


Fig.3.7. Hardware implementation of multiplication with constant φ.

I. Multiplicative Inversion in GF(24):

$q_3^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_0 \oplus q_2$............ (14)

$q_2^{-1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1$......... (15)

$q_1^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_1q_0 \oplus q_2 \oplus q_2q_0 \oplus q_1$.......(16)

$q_0^{1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_1 \oplus q_3q_1q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1 \oplus q_2q_1q_0 \oplus q_1 \oplus q_0$............... (17)

IV. S-BOX USING PIPELINING

Pipelining originates from the idea of a water pipe with continuous water sent in without waiting for the water in the pipe to come out. Accordingly, it results in speed increases. Pipelining is an implementation technique where multiple instructions are overlapped in execution. The computer pipeline is divided in stages. Each stage completes a part of an instruction in parallel. The stages are connected one to the next to form a pipe - instructions enter at one end, progress through the stages, and exit at the other end [7].

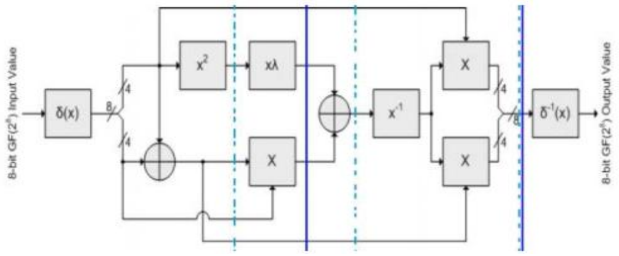


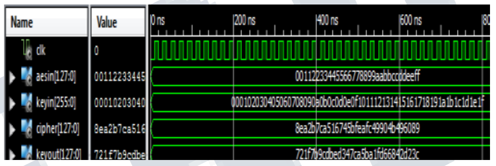
Fig.4.1 The places of two and three pipeline stages S-Box

The different stages of pipelined implementations having S-Box, the most resourceful and efficient designs are those with two and three pipeline stages for the byte substitution phase as shown in fig. Here the dotted lines are the pipel registers for the three-stage S-Box and the solid lines are the registers for two-stage S-box[6][2].

V. SIMULATION WAVE FORMS AND RESULTS

AES-256 Encryption simulation:

In the Encryption process we can consider a plaintext of 128 bits is "00112233445566778899aabbccddeeff" and key of 256 bits is "000102030405060708090a0b0c0d0e0f101112131415 161718191a1b1c1d1e1f".

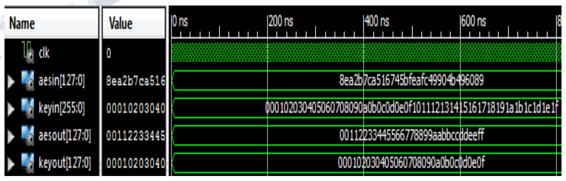


```
Name           Value         0 ns      200 ns      400 ns      600 ns      80
clk            0
aesin[127:0]   00112233445   00112233445566778899aabbccddeeff
keyin[255:0]   00010203040   000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
cipher[127:0]  8ea2b7ca516   8ea2b7ca516745bfeafc49904b496089
keyout[127:0]  721f7b9cdbe   721f7b9cdbed347ca5ba1fd66842d23c
```

Fig.5.1 Encryption simulated wave form

AES-256 Encryption simulation:

In the Decryption process we can consider input as a output of encryption process of cipher text of 128 bits is "8ea2b7ca516745bfeafc49904b496089 " and key of 256 bits is "000102030405060708090a0b0c0d0e0f101112131415 161718191a1b1c1d1e1f".



```
Name           Value         0 ns      200 ns      400 ns      600 ns      8
clk            0
aesin[127:0]   8ea2b7ca516   8ea2b7ca516745bfeafc49904b496089
keyin[255:0]   00010203040   000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
aesout[127:0]  00112233445   00112233445566778899aabbccddeeff
keyout[127:0]  00010203040   000102030405060708090a0b0c0d0e0f
```

Fig.5.2 Decryption simulated wave form

VI. CONCLUSION

The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. The number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit key, 12 rounds for 192-bit key and 14 rounds for 256-bit key. When a key of size 256 bits is used, the number of rounds are repeated is equal to 14. The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. It substitutes all bytes of the State using a look-up table called S-Box. A more suitable method of

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 3, March 2017

implementing the S-Box is to use composite field approach for calculating multiplicative inverse in a Galois Field *GF(2^8)* followed by an affine transformation in the binary extension field , which is based on combinational logic. An efficient design is three stages pipelining for the byte substitution phase. The proposed Three Stages Pipelining S-Box Using Logic gates simulated and synthesized by using Xilinx 14.7 tool.

REFERENCES

[1] FIPS 197, "Advanced Encryption Standard (AES)", November 26, 2001.

[2] Marko Mali, Franc Novak and Anton Biasizzo "Hardware Implementation of AES Algorithm" – Journal of ELECTRICAL ENGINEERING, Vol. 56, No. 9-10, 2005, 265-269.

[3] Behrouz A. Forouzan and Debdeep Mukhopadhyay "Cryptography and Network Security" (2nd edition).

[4] L.Thulasimani,"A Single Chip Design and Implementation of AES-128/192/256 Encryption Algorithms"- International Journal of Engineering Science and Technology, Vol. 2(5), 2010, 1052-1059.

[5] Nation Institute of Standards and Technology (NIST), Data Encryption Standard (DES), National Technical Information Service, Sprinfgield, VA 22161, Oct. 1999.

[6] J. Daemen and V. Rijmen, "AES Proposal: Rijndael", AES Algorithm Submission, September 3, 1999.

[7] J. Nechvatal et. al., Report on the development of Advanced Encryption Standard, NIST publication, Oct 2, 2000.

[1]**Pujari Munaswamy** did his B.Tech in Eelectronics and Communicatoin Engineering at Nimra College of Engineering and Technology, Vijayawada and doing Master of Technology in Digital Electronics and Communication Systems at Annamacharya Institute of Technology and Sciences, Tirupati.

[2]**Mr.N.Dilipkumar** obtain his B.Tech(ECE) at N.B.K.R.I.S.T, Vidyanagar in 2010 and Master degree from SRM University, Chennai in 2014 and his area of interest is testing of VLSI Circuits. He is having 3Years of teaching experience. He is currently working as Assistant professor, in Annamacharya Institute of Technology and sciences, Tirupati. He has been active in research and published 3 international journals, 1 Internatioinal Conference & 2 National conferences.

[3]**Ms.N.Pushpalatha** completed her B.Tech at JNTU, Hyderabad in 2004 and M.Tech at A.I.T.S., Rajampet in 2007. Presently she is working as Assistant Professor of ECE, Annamacharya Institute of Technology and Sciences, Tirupati since 2006. She has guided many B. Tech projects. Her Research area includes Data Communications and Ad-hoc Wireless Sensor Networks.