ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

Design of Least Complex S-Box and its Fault Detection for Robust AES Algorithm

G. Alisha Evangeline, S. Krithiga, J. Jesu Mejula

***Abstract*— Advanced Encryption Standard (AES) is the symmetric key standard for encryption and decryption. In this work, a 128-bit AES encryption and decryption using Rijndael Algorithm is designed and synthesized using verilog code. The fault detection scheme for their hardware implementation plays an important role in making the AES robust to the internal and malicious faults. In the proposed AES, a composite field S-Box and inverse S-Box is implemented using logic gates and divided them into five blocks. Any natural or malicious faults which defect the logic gates are detected using parity based fault detection scheme. For increasing the fault exposure, the predicted parities of each of the block S-box and inverse S-box are obtained. The multi-bit parity prediction approach has low cost and high error coverage than the approaches using single bit parities. The Field Programmable Gate Array (FPGA) implementation of the fault detection structure has better hardware and time complexities.**

***Index Terms*— AES Algorithm, composite field S-box, error coverage, Galois Field, parity based fault detection.**

I. Introduction

The data encryption is the process of transforming data (plain text) using an algorithm called cipher to make it unreadable to anyone except those possessing special knowledge usually referred to as key. The encryption algorithm performs various substitutions and transformations on the plain text. There are two encryption standards namely Data Encryption Standard (DES) and Advanced Encryption Standard (AES). The Data Encryption Standard (DES) is a 64-bit cipher where the data are encrypted in 64-bit blocks using a 56-bit key. The DES algorithm transforms 64-bit input in a series of steps into a 64-bit output. There are two inputs to the encryption function: the plain text to be encrypted and the key. On 2nd January, 1997, the National Institute of Standards and Technology (NIST) invited proposals for new algorithms for the new Advanced Encryption Standard (AES). The goal was to replace the older Data Encryption Standard (DES) which was introduced in November, 1976 as it was no longer secure. After going through two rounds of evaluation, the two Belgian Cryptographers Joan Daemen and Vincent Rijmen (originally called Rijindael) were selected and named the Advanced Encryption Standard on 26th November, 2001. The Advanced Encryption Standard (AES) supercades the Data Encryption Standard (DES). The algorithm described by AES is a symmetric key algorithm meaning the same key is used for encrypting and decrypting the data. AES is based on the design principle known as substitution permutation network and it is fast in both software and hardware. AES is a modification of Rijndael which has a fixed block size of 128 bits and a key size of 128,192 or 256 bits. It operates an 4x4 column major order matrix of bytes termed the state although some version of Rijndael have a larger block size have additional columns in the state. Most AES transformations are done in a special finite field. In the AES, the cipher text is generated after 10 rounds where encryption round consists of 4 transformations which are add round key, sub bytes, shift row and mix column transformation. The decryption algorithm transforms the cipher text to the original plain text using the reverse procedure. In AES transformation, only the s-boxes in the encryption and inverse s-boxes in decryption are non linear. The transformations occupy much of the total AES encryption/decryption area. The receiver with a specific key would only be able to retrieve the original data when using AES in transferring data but reliability of data transfer is not guaranteed because of defects in implemented structure of AES or interventions of attackers. Fault detection is an inevitable part of AES hardware implementation because of two reasons: Natural faults caused by defects in gates may result in errorneous output in encryption/decryption. Attackers can also inject certain faults in AES to retrieve the key & break the system. There exist many schemes for detecting the faults in the hardware implementation of the AES. They are concurrent Error Detection Scheme, Incorporating Error Detection, Double Data Rate Computation and Differential Fault Analysis.

.

II. Related Works

In [1], one of the best symmetric security algorithms is used to provide data security in AES. The pipelined architecture of the AES algorithm increases the throughput of the algorithm and the pipelined key schedule algorithm increases the speedup. In this architecture, instead of passing the output of each round to the next round directly, a register is used. It avoids the direct contact between two rounds. With the help of search based Look-Up-Table, the hardware cost is reduced. The speed of the AES algorithm is increased by inserting compact and flexible architecture for Mix Column transform. In [2], a fixed coefficient multiplier for Mix Column operation and an equivalent pipelined architecture leads to effective utilization of resources and increase in speed. The modifications in each round of the AES algorithm in [3],

G. Alisha Evangeline, *M.E Applied Electronics, Jerusalem College of Engineering, Chennai, India, Mobile No. 7708702692*
S. Krithiga, *ECE Department, Jerusalem College of Engineering, Chennai, India, Mobile No. 9710907221,*
J Jesu Mejula, *ECE Department, Jerusalem College of Engineering, Chennai, India, Mobile No. 9444312365*

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

improved the complexity of the encryption method and making it complicated for the attacker to predict a pattern in the algorithm. In each transformation of the modified architecture, the 8-bit values are separated in to 4-bits and they are grouped and then perform the transformation process. The modifications have provided the algorithm with strong diffusion and confusion. A high data throughput AES hardware architecture is proposed in [4] by partitioning the 10 rounds into sub blocks of repeated AES modules. To provide a complete ten stages of AES, the intermediate buffers are used to separate the blocks. Using this pipelined architecture scheme, time complexity is reduced to greater extent. In [5], a simple, linear and cryptanalysis is done on the standard S-Box to take advantage of high probability occurrences of linear expressions involving plain text bits , cipher text bits and sub-key bits. The operation of the cipher is linear where the linearity refers to a mod-2-bitwise operation. The design can run at 1.2 GHz which is sufficient for online data encryption. The Mix Column in [6] could be designed easily using one basic module which imposes one time block, two or three byte-XOR logic and additional data path selector. The optimized architecture of data encryption unit and key schedule unit is applicable to wireless sensor networks. In [7], a 128-bit AES encryption and decryption using Rijndael Algorithm is designed and synthesized using verilog code which can be easily implemented with the help of FPGA. The design and performance testing algorithm in [8] is implemented with the help of dynamic partially reconfigurable FPGA. To self select the coprocessors, a FPGA based Micro-Blaze processor is used which reduces area requirements and increase system's versatility. To increase the performance of the executed circuit, particularly cost and power, all of the AES blocks may be reconfigurable. So the parameters used for reconfiguration are implanted inside the manager module of reconfiguring, and also possible to quickly cross from a safe configuration to another by updating a hard system protection.

III. Proposed System

Using composite fields, we introduce a low cost multiple parity based fault detection scheme for the s-box and inverse s-box. The s-box and inverse s-box are non-linear operations which take 8 bit input and generate 8 bit output. Normally, the s-box is implemented as ROM, there is a possibility that the hackers may hack the confidential data. Here we implement the s-box in hardware level where hacking is not possible. To increase the error coverage, the s-box and inverse s-box are divided into 5 blocks where 3 predicted parities are used for multiplicative inversion and the rest for the affine matrix and transformation. Each block is compared with the reference circuit. The optimum S-boxes and inverse S-boxes using normal basis are more compact than the one using polynomial basis. However, using polynomial basis, results in fast implementation. The implementation is done in verilog code and dumped in FPGA kit. The input is given to the s-box and the outputs are tested with the reference circuit. Both the hardware and time complexities are reduced to maximum extent.

A. AES Algorithm

The AES Algorithm operates on a 4x4 array of bytes which is called a state. The state undergoes four transformations namely Sub Bytes, Shift Rows, Mix Columns and Add Round Key. The AES Encryption Round is shown in fig. 1.

B. Sub Bytes and Inverse Sub Bytes

The first transformation in each round is the bytes substitution called Sub Bytes which is implemented by 16 S-boxes. It is a nonlinear substitution step where each byte is replaced with another according to the look-up table. Each s-box transformation performs multiplicative inversion for numbers 00H-FFH in GF (2^8) followed by an affine transformation. For inverse S-box transformation, the inverse affine transformation takes place first prior to computing the multiplicative inverse. The individual bits in a byte representing GF (2^8) elements can be viewed as coefficients to each power term in the GF (2^8) polynomial. The S-box and inverse S-box of the AES is divided in to five blocks and the predicted parities of these blocks are calculated. Out of these five blocks, three blocks perform multiplicative inversion and the remaining performs the transformation and affine matrices, based on Galois field operation. The block diagram of S-box and inverse S-box using polynomial basis is shown in fig.2.

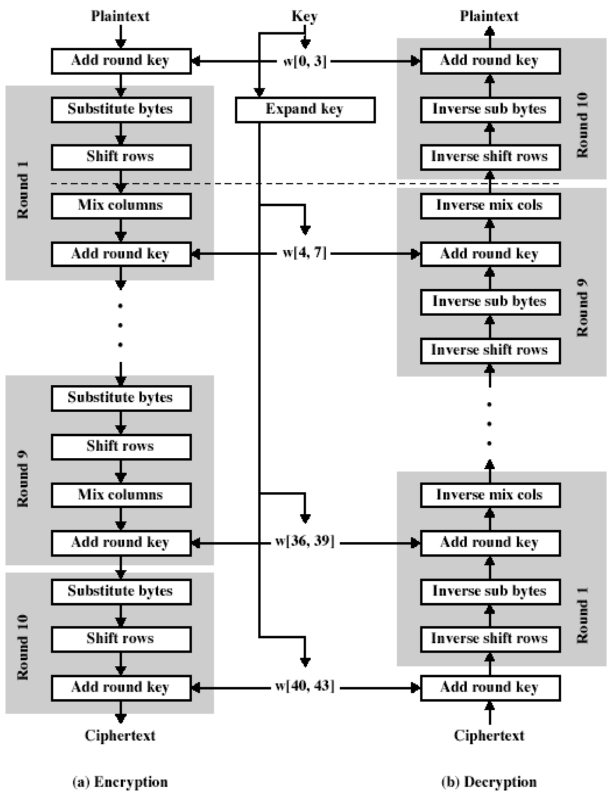


Fig.1 Overall Structure of AES Algorithm

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

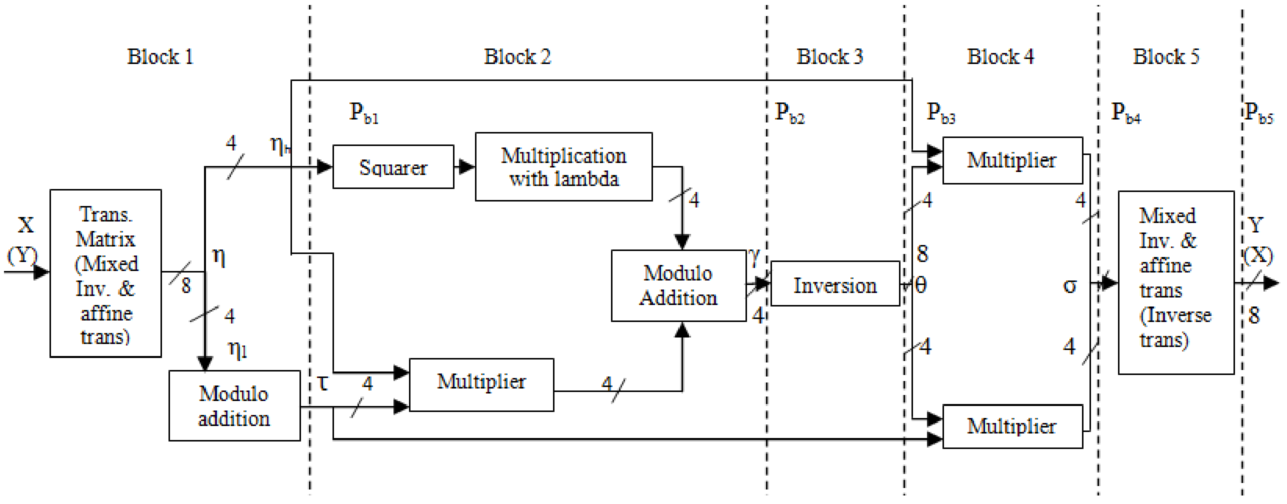


Fig.2. Block diagram of S-box and inverse S-box using polynomial basis

C Shift Rows And Inverse Shift Rows

Shift Rows is a transposition step where each row of the state is shifted cyclically a certain number of steps to left. For AES, the first row is left unchanged. In the second row, each byte is shifted one position to the left. Similarly, the bytes in the third and fourth rows are shifted by two and three positions respectively. The shift rows transformation is shown in fig. 3. Inverse shift rows is the inverse process of Shift rows transformation in which the bytes in the last three rows of the State are cyclically shifted over different numbers of steps to right. For AES decryption, the first row, ie, $r = 0$, is not shifted. In the second row, each byte is shifted one position to the right. Similarly, the bytes in the third and fourth rows are shifted by two and three positions respectively.

D. Mix Column and Inverse Mix Column

In Mix Columns, each entry in the output state is constructed by the multiplication of a column in the input state with a fixed polynomial over GF(2^8). The Mix column is designed using one basic module, which contains one xtime block, XOR logics and data path selector. The Mix column and its basic module are shown in fig.4. The dashed line represents the basic module. In Mix column, the xtime module can be implemented with the combination of XOR gates and logic shift operations. The Mix column transformation is executed by repeating the operation of basic module into four times. Inverse Mix column is the inverse operation of the Mix column transformation.

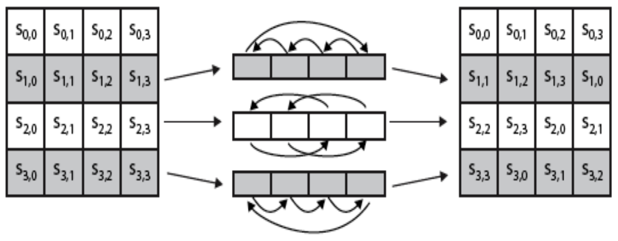


Fig.3. Transformation in shift rows

E. Add round key

In this, each byte of the state is combined with the round key: each round key is derived from the cipher key using a key schedule. The round key is added to the state before starting the loop. In the Add Round Key step, each byte of the key state is combined with a byte of the round sub key using the XOR operation. The Add Round Key transformation is given in fig. 5.

F. AES Key Expansion

The AES key expansion algorithm takes a 4-word key as input and produces a linear array of 44 words. Each round uses 4 of these words. Each word contains 32 bytes which means each sub key is 128 bits long. Rot Word performs a one-byte circular left shift on a word. This means that an input word [a0, a1, a2, a3] is altered into [a1, a2, a3, a0]. Sub Word performs a byte substitution on each byte of its input word, using the s-box. Then the outcome of rot word and sub word is added using XOR operation with round constant, Rcon[j]. The round constant (Rcon[j]) represents a word in which the three rightmost bytes are always 0.

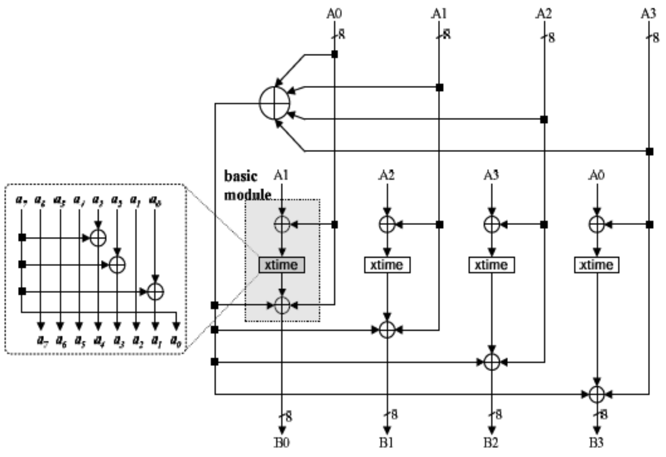


Fig.4. Mix Column and its Basic Module

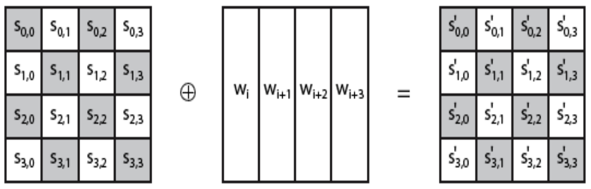

$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
⊕
w_i w_{i+1} w_{i+2} w_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Fig.5. Transformation in Add Round Key

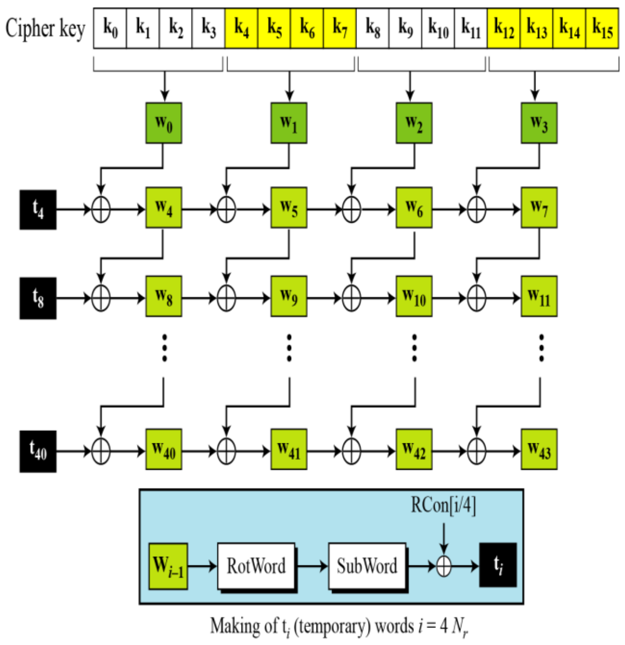

Cipher key
k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10} k_{11} k_{12} k_{13} k_{14} k_{15}
w_0 w_1 w_2 w_3
t_4 w_4 w_5 w_6 w_7
t_8 w_8 w_9 w_{10} w_{11}
t_{40} w_{40} w_{41} w_{42} w_{43}
RCon[i/4]
w_{i-1}
RotWord
SubWord
t_i
Making of t_i (temporary) words $i = 4\,N_r$

Fig.6. Key Expansion in AES

Thus the effect of an XOR operation of a word with Rcon is to only perform an XOR operation on the leftmost byte of the word. The round constant (Rcon[j]) is different for each round and is defined as Rcon[j] = (RC[J], 0,0,0), with RC[1]= 1, RC[j]= 2* RC[j − 1] and with multiplication defined over the field GF(2^8). The key expansion was designed to be resistant to know the attacks in cryptanalysis. The accumulation of a round-dependent round constant eliminates the symmetry between the ways in which round keys are generated in different rounds. The generation of expanded key from the actual key is viewed in fig.6.

IV. Fault Detection Scheme

There are many fault detection schemes [9], [10] designed for detecting faults in Substitution Bytes transformation of the AES. In order to achieve a high performance AES, a multiple bit parity based fault detection scheme is used. In this method, the S-box and inverse S-box of the AES is divided in to five blocks and the predicted parities of these blocks are calculated. The operations of each block can be explained as follows:

A. Blocks 1 and 5

In fig.3, blocks 1 and 5 of the S-box and inverse S-box consist of transformation and inverse transformation matrices as well as affine and inverse affine transformation. The elements in the transformation and inverse transformation matrices can be mapped using isomorphic and inverse isomorphic function. Let us consider X is the input of the S-box. The predicted parity of blocks 1 and 5 in S-box and inverse S-box are denoted as $\hat{P}_{b1}$ and $\hat{P}_{b5}$. The predicted parities are given below in (1).

$$\hat{P}_{b1} = X[0] + X[2] + X[4] + X[5]$$

$$\hat{P}_{b5} = \sigma[0] + \sigma[1] + \sigma[2] + \sigma[4] + \sigma[6]$$

(1)

Where, + performs modulo-2-addition operation.

B. Blocks 2 and 4

As shown in fig.3, block 2 of S-box and inverse S-box contains multiplication, addition and squaring operation. These operations are performed by Galois Field using polynomial basis. Using polynomial expression, the area of the S-box and inverse S-box is reduced. GF (2^4) multiplication takes place in block4. The predicted parities of block 2 and 4 in S-box and inverse S-box are denoted as $\hat{P}_{b2}$ and $\hat{P}_{b4}$. They are expressed in (2) as follows:

$$\hat{P}_{b2} = \eta[4] + \eta[3]\,((\sim p_H) + \eta[6]) + (\sim\eta[2](p_H + \eta[6]) + \eta[1](\eta[6] + \eta[4]) + \eta[0](\sim P_H)$$

$$\hat{P}_{b4} = \eta[3](\theta[0] + \theta[2] + \theta[3]) + \eta[2](\theta[0] + \theta[1] + \theta[3]) + \eta[1](\theta[6] + \theta[4]) + \eta[0](\theta[0] + \theta[1] + \theta[2] + \theta[3]) \quad (2)$$

Where, $P_H = \eta[7]+\eta[6]+\eta[5]+\eta[4]$, ~ denotes inverse operation, + performs modulo-2-addition operation and □ represents OR operation.

C. Block 3

As mentioned in fig.3, block 3 of the S-box and inverse S-box performs multiplicative inverse operation. It accepts 4-bit input and gives 4-bit output. The predicted parity of block 3 is denoted as $\hat{P}_{b3}$. It is given in (3) as,

$$\hat{P}_{b3} = (\gamma_1 + \gamma_0)\,\gamma_3 + (\overline{\gamma}_2 \;\square\; \gamma_1)\,\gamma_0$$

(3)

Where, + performs modulo-2-addition operation and ^ represents OR operation.

The predicted parities of each block can be calculated using eqns. (1), (2) and (3). Then the predicted parities are compared with the actual predicted parities. The actual predicted parities are precised in eqn. (4).

$$P_{b1} = \sum\nolimits^{3}_{i=0} \tau_i$$

$$P_{b2} = \sum\nolimits^{3}_{i=0} \gamma_i$$

$$P_{b3} = \sum\nolimits^{3}_{i=0} \theta_i$$

$$P_{b4} = \sum\nolimits^{7}_{i=0} \sigma_i$$

$$P_{b5} = \sum\nolimits^{7}_{i=0} y_i \qquad (4)$$

Where, ∑ denotes modulo-2-addition operation.

The faults in the S-box and inverse S-box can be detected by comparing the predicted parity of each blocks using XOR operation. The error indication flags are used to represent fault detection. For error free computations, all the five error indication flags should be zero. They are expressed in (5) as follows:

$$E1=P^{\wedge}{}_{b1} + P_{b1}$$
$$E2=P^{\wedge}{}_{b2} + P_{b2}$$
$$E3=P^{\wedge}{}_{b3} + P_{b3}$$
$$E4=P^{\wedge}{}_{b4} + P_{b4}$$
$$E5=P^{\wedge}{}_{b5} + P_{b5} \qquad (5)$$

Where, + performs modulo-2-addition operation.

V. Implementation Results

The S-box and inverse S-Box are designed using logic gates for reducing the hardware complexity and detecting the faults in it. The multiple faults can be applied to the S-box and inverse S-box and the output is noted. Then the S-box and inverse S-box is tested using FPGA kit. The simulated result can be obtained using ModelSim SE plus 6.4c and synthesis is carried out using Xilinx ISE. Then the delay and error coverage can be calculated using the simulated results. They are explained below.

A. *Delay and Area*

The number of LUTs and slices used to design the S-box and inverse S-box is calculated from the simulation results. Table-I represents the comparison of the number of LUTs and slices used for designing the S-box and inverse S-box. From Table-I, the proposed S-box contains less number of LUTs and slices when compared to low power S-box and S-box using LUTs. The gate delay and net delay of the proposed S-box and inverse S-box are given in Table-II.

B. *Error Coverage*

The proposed S-box and inverse S-box is able to find the Stuck-at faults and random faults in the circuit. The faults are injected in any blocks of the S-box and inverse S-box using logic gates. In case of multiple faults injected in S-box, 100 faults have been injected in the circuit. Out of these, 99 faults have been recognized. In case of multiple faults in inverse S-Box, out of 100 faults 99 have been recognized.

TABLE-I
COMPARISON OF LUTS AND SLICES

	No. of 4 input LUTs	No. of slices
LUT based S-box	250	158
LUT based inverse S-box	250	158
Composite S-box	83	43
Composite inverse S-box	73	38
Low power S-box	87	46
Low power inverse S-box	84	44
Proposed S-box	71	41
Proposed inverse S-box	69	39

TABLE-II
GATE DELAY AND NET DELAY

	GATE DELAY	NET DELAY	TOTAL DELAY
PROPOSED S-BOX	11.851 NS	12.988 NS	24.839 NS
PROPOSED INVERSE S-BOX	11.581 NS	12.008 NS	23.589 NS

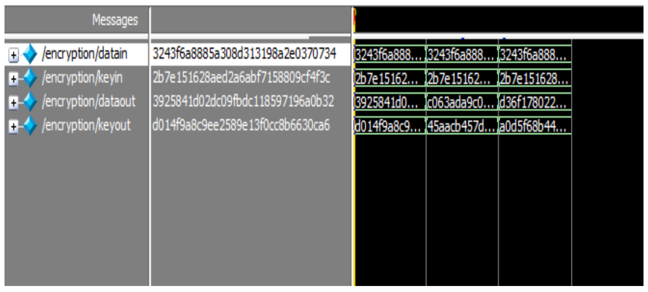
TABLE-III
ERROR COVERAGE

FAULTS	ERROR COVERAGE
SINGLE FAULT IN S-BOX	100%
SINGLE FAULT IN INVERSE S-BOX	100%
MULTIPLE FAULTS IN S-BOX	99%
MULTIPLE FAULTS IN INVERSE S-BOX	98%

Table-III represents the error of S-box and inverse S-box. Using the simulation results, the error coverage is calculated. From Table-III, the error coverage in S-box and inverse S-box is approximately 99.25%.

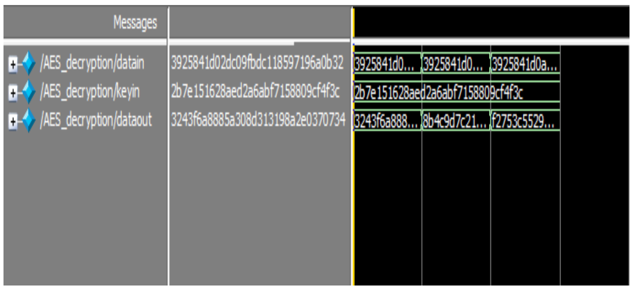
C. *Simulation Result*

The S-box and inverse S-box are designed and simulated using Modelsim. The simulated result for the S-box and inverse S-box is revealed in fig. 7 and fig. 8.



```
Messages
/encryption/datain    3243f6a8885a308d313198a2e0370734   3243f6a888...  3243f6a888...  3243f6a888...
/encryption/keyin     2b7e151628aed2a6abf7158809cf4f3c   2b7e15162...   2b7e15162...   2b7e151628...
/encryption/dataout   3925841d02dc09fbdc118597196a0b32   3925841d0...   c063ada9c0...  d36f178022...
/encryption/keyout    d014f9a8c9ee2589e13f0cc8b6630ca6   d014f9a8c9...  45aacb457d...  a0d5f68b44...
```

Fig.7. Simulated output for AES Encryption



```
Messages
/AES_decryption/datain    3925841d02dc09fbdc118597196a0b32   3925841d0...   3925841d0...   3925841d0a...
/AES_decryption/keyin     2b7e151628aed2a6abf7158809cf4f3c   2b7e151628aed2a6abf7158809cf4f3c
/AES_decryption/dataout   3243f6a8885a308d313198a2e0370734   3243f6a888...  8b4c9d7c21...  f2753c5529...
```

Fig.8. Simulated output for AES Decryption

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

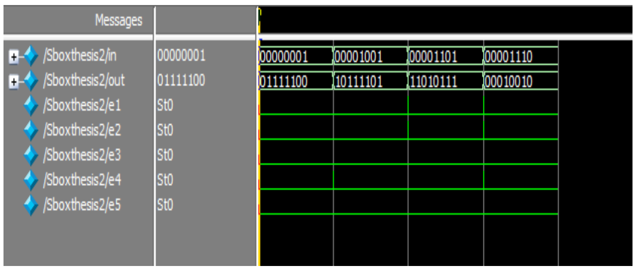


Fig.9. Simulated output for Fault Detection in S-box

The simulated output for fault detection in S-box when there is no fault in the circuit is as shown in fig.9. From the figure, we conclude that there is no fault in the S-box circuit where all the error indication flag of the five blocks in the S-Box is zero.

VI. CONCLUSION

In this paper, the 128-bit AES encryption and decryption is designed and synthesized using verilog codes in ModelSim. The S-box and inverse S-box in AES has been designed using logic gates. A multiple bit parity based fault detection scheme for the AES using composite field S-box and inverse S-box is accessible in order to diagonise the faults in the hardware implementation of S-box and inverse S-box. The simulation results of the fault detection based scheme S-box and inverse S-box has high error coverage when compared to other fault detection schemes. Also this scheme has less hardware and time complexities and the results are given above.

REFERENCES

[1] Subashri T, Arunachalam R, Gokul Vinoth Kumar B, and Vaidehi V, "Pipelining Architecture of AES Encryption and Key Generation with Search Based Memory," *International journal of VLSI design & Communication Systems (VLSICS)*, Vol.1, No.4, December 2010.

[2] J. Vijaya and M. Rajaram, "High Speed Pipelined AES with Mix Column Transform," *European Journal of Scientific Research*, ISSN 1450-216X Vol.61 No.2 (2011), pp. 255-264.

[3] Priyanka Pimpale, Rohan Rayarikar, Sanket Upadhyay, "Modifications to AES Algorithm for Complex Encryption," *IJCSNS International Journal of Computer Science and Network Security*, Vol.11 No.10, October 2011.

[4] Ahmed. H. Sawalmeh, "Hardware Design of AES S-box using pipelining structure over GF($(2^4)^2$)".

[5] K.Rahimunnisa, Dr. S. Sureshkumar, and K.Rajeshkumar, "Implementation of AES with New S-Box and Performance Analysis with the Modified S-Box," *International Conference on VLSI, Communication & Instrumentation (ICVCI) 2011 Proceedings published by International Journal of Computer Applications® (IJCA).*

[6] MooSeop Kim, Juhan Kim, and Yongje Choi, "Low Power Circuit Architecture of AES Crypto Module for Wireless sensor Network," *World Academy of Science, Engineering and Technology* 8, 2007.

[7] M.Pitchaiah, Philemon Daniel, and Praveen, "Implementation of Advanced Encryption Standard Algorithm," *International Journal of Scientific & Engineering Research*, Volume 3, Issue 3, March -2012 1 ISSN 2229-5518.

[8] Zine El Abidine, Alaoui Ismaili, and Ahmed MOUSSA, "Self-Partial and Dynamic Reconfiguration Implementation for AES using FPGA," *IJCSI International Journal of Computer Science*, Issues, Vol. 2, 2009 ISSN (Online): 1694-0784 ISSN (Print): 1694-0814.

.

G.Alisha Evangeline received her B.E degree in Electronics and Communication Engineering at C.S.I Institute of Technology, Thovalai, India in the year, 2011. She is currently persuing her M.E degree in Applied Electronics at Jerusalem College of Engineering, Chennai, India. She has published five papers in National & International conferences. Her area of research includes Cryptography, VLSI Testing and Embedded Systems.

Mrs. S. Krithiga received her B.E Electronics and Communication Engineering from University of Madras in the year 2000 and M.E in College of Engineering, Guindy Campus, Chennai, India. She is presently working at Jerusalem College of Engineering, Chennai, India. She has published three papers in National & International conferences. Her research interests are VlSI Design, Cryptography and Hardware and Software co-design.

Mrs. J. Jesu Mejula received her B.E Electronics and Communication Engineering from Noorul Islam College of Engineering, Thukkalai, Nagerkoil, India in the year 2000 and M.E in College of Engineering, Guindy Campus, Chennai, India. She is presently working at Jerusalem College of Engineering, Chennai, India. She has published five papers in National & International conferences. Her research interests are Microprocessors, Embedded System and Digital Image processing.