 ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Advanced Encryption Standard and Fault Detection for High Performance

[1]Jishamol T.K, [2]K. Rahimunnisa, [3]Ginu Thomas
[1,2,3]School of Electrical, Sciences, Karunya University, Coimbatore, Tamilnadu, India

Abstract
Advanced Encryption Standard (AES) algorithm is a symmetric block cipher that can encrypt and decrypt the information. This paper presents the AES Algorithm. The look up table is used for implementing the AES S-box and inverse S-box. The AES has been built as the first choice for many cryptographic applications because of the high level of security. However the faults that can accidently occur in the hardware implementation may cause erroneous encrypted or decrypted output that, results in losing the original information. At the end of this design, a parity based fault detection scheme for high performance AES is also presented.

Keywords
Advanced Encryption Standard (AES), S-Box, Parity Prediction, Fault Detection

I. Introduction

Cryptography is the practice and study of hiding information, which plays an important role in the security of data transmission. Apart from its uses in Military and Government to facilitate secret communication, Cryptography is used in protecting many kinds of civilian systems [1]. The Encryption algorithm is a set of well defined steps to transform data from a readable format to an encoded format using the key. This algorithm provides Confidentiality, Authentication and integrity. Encryption algorithms are broadly classified as Symmetric or Asymmetric algorithms based on the kind of keys used. For symmetric algorithm the Encryption and Decryption uses same key. A few well-known examples are: DES, AES. Asymmetric algorithm uses separate keys for both Encryption and Decryption. The decryption key is typically kept secretly, therefore called private key, while the encryption key is spread to all who might want to pass the encrypted messages, therefore called public key. Symmetric algorithms have the advantage of not consuming too much computing power. Advanced Encryption Standard is one of the Symmetric Algorithm [2].

Advanced Encryption Standard, based on the Rijndael algorithm is such a Symmetric algorithm for Encryption. The Advanced Encryption Algorithm (AES) was published by NIST (National Institute of Standard and Technology) in 2001 [2]. NIST selected Rijndael as the proposed AES algorithm. Rijndael has many advantages. It has resistance against all known attacks. The design is simple. It has high speed and the code is compact. AES is a symmetric block cipher that replaced DES. AES has a wide range of-application. The AES has a block length of 128 bits and key length of 128,192 or 256 bits. The basic unit of processing in the AES algorithm is a byte. The AES operates on a 4x4 array of bytes which is called a state. The state undergoes 4 transformations which are namely the AddRoundKey, SubByte, ShiftRow and MixColumns transformations. Among these transformations in the AES, the S-Box and inverse S-Box are alone nonlinear and these occupy much of the total AES area. Fault detection scheme in the AES implementation is important in order to make the standard robust to internal faults. There exist many fault detection schemes for the AES implementations. For the fault detection in the AES, redundant units may be used [4-5]. The fault detection is achieved using LUTs in [6], where as the fault detection in [7] uses memories. A fault detection scheme based on hamming codes for protecting storage elements is presented in [8].

The remainder of this paper is organized as follows. Section II briefly describes the AES algorithm. Section III explains the fault detection scheme. The results and discussions are given in Section IV. The paper ends by drawing some conclusions in Section V.

II. Aes Algorithm

The AES has been fully documented in the freely available U.S government publication Federal Information Processing Standards (FIPS) 197. This cipher can perform encipher and decipher operations. The AES algorithm operates on 128 bits of data and generates 128 bits of output. The length of key used to encrypt this input data can be 128, 192 or 256 bits. Typically 128 bit key is used for both encryption and decryption. As this is a private key cipher it uses just same key of 128 bits which is used for both encryption and decryption. AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. Figure 1 shows the AES encryption. The first round comprises a 128-bit XOR of the plaintext with the key to form a new 128-bit state. Each middle round operates on the states by performing the operations ShiftRows, SubBytes, MixColumns, and AddRoundKey. The final round is similar to the middle rounds except that there is no MixColumns step. Figure 2 shows the decryption algorithm. The decryption is the inverse of encryption. The last decryption iterations exclude the Inverse MixColumn Steps [2].

AES defines a 16×16 matrix of byte values, called an S-box. SubByte transformation is a nonlinear operation that works on individual bytes using a substitution table, which contains a permutation of all 256 possible 8-bit values. Each individual byte of state is mapped into a new byte in the following manner: The left most 4bits of the byte are used as a row value and the rightmost 4 bits are used as column value. These row value and column value serve as indexes into the S-Box to select a unique 8-bit output value. Shift Row transformations essentially consists of shifting the bytes in the row. It is a transposition step on the row of the state where each row of the state is shifted cyclically by certain number of steps. The first row (row 0) is unaltered. The second row is shifted by one byte, the third row is shifted by two bytes and final row is shifted three bytes. The MixColumns function takes four bytes as input and outputs as four bytes, where each input byte affects all four output bytes. Together with the ShiftRows, MixColumns provides diffusion in the AES cipher. Each column in MixColumn is treated as a polynomial over $GF(2^8)$ and is then multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x) = \{03\}x^3 + \{01\}\ x^2 + \{01\}\ x + \{02\}$. During this operation, each column is multiplied by the known-matrix. In AddRoundKey transformation, a Round Key is added to the State value by a simple XOR operation. Each Round Key involves the words from the key schedule. Those words are added into the columns of the state.

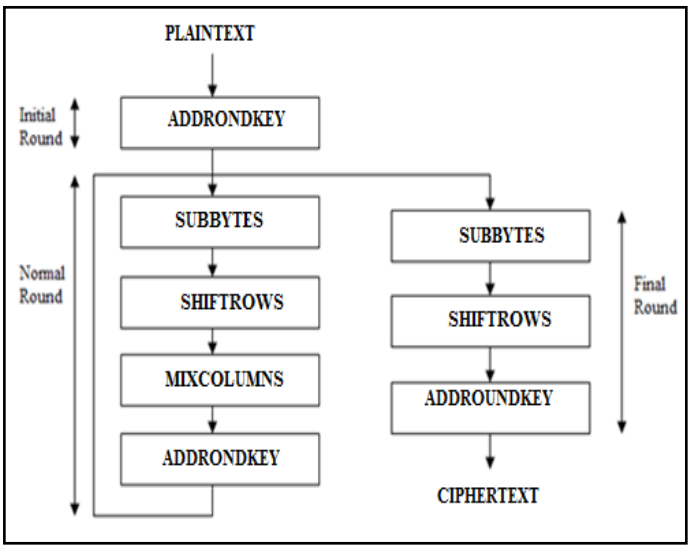

Fig. 1: 128-bit AES Encryption

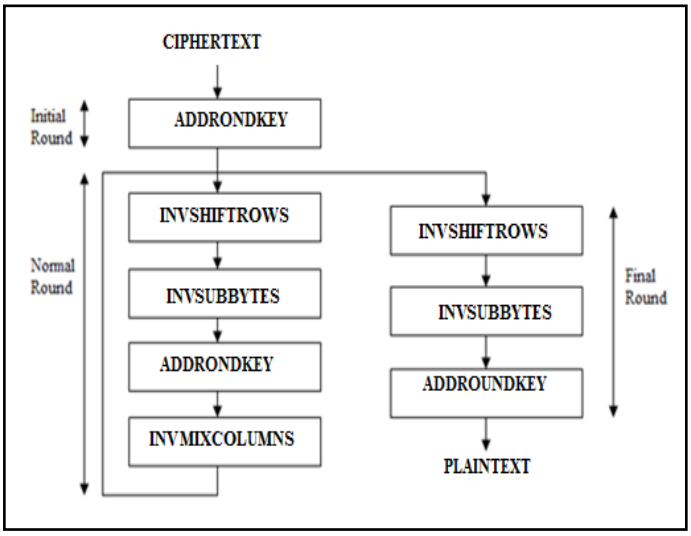

Fig. 2: 128-bit AES Decryption

The AES algorithm takes as input a 4-word key and produces a linear array of 44 words. This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The key is copied in to first four words of the expanded key. The remainder of the expanded key word is filled in four words at a time. The key expansion has three steps: Byte Substitution subword, Rotation rotword and XOR with RCON (round constant) as in fig. 3.

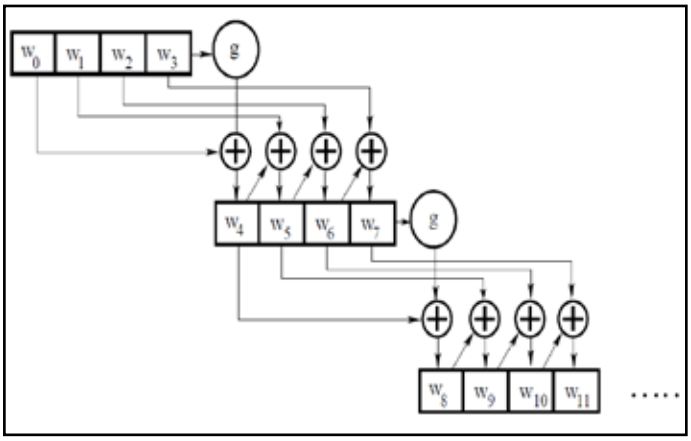

Fig. 3: Key Expansion

Key expansion generates a total of $N_b(N_r + 1)$ words. The algorithm requires an initial set of N_b words, and each of the N_r rounds requires N_b words of key data. The resulting key schedule involves a linear array of 4-byte words, denoted $[w_i]$, with i in the range $0 \leq i < N_b(N_r+1)$.The subword function takes a four byte input and applies the byte substitution operation and produces an output word. The rotword takes a word $[w_{00},w_{01},w_{02},w_{03}]$ as input and performs a cyclic permutation to produce $[w_{01},w_{02},w_{03},w_{00}]$ as output word. SubWord (RotWord (temp)) is XORed with Rcon[j] (round constant). The round constant is a word in which the three rightmost bytes are zero. It is different for each round and which is defined as: Rcon[j] = (RC[j],0,0,0), where RC[1] = 1, RC[j]= 2 * RC[j-1]. Multiplication is defined over Galois Field. Values of RC[j] in hexadecimal are shown in Table 1 [3].

Table 1: Rcon Values

J	1	2	3	4	5	6
RC[j]	01	02	04	08	10	20

III. Fault Detection Scheme

The AES is the private-key cryptographic algorithm used for transferring block of data more securely. However the faults that accidently occur on AES may cause erroneous encrypted or decrypted output that result in losing the original information. In this paper parity based fault detection schemes for high performance AES implementations is presented. The objective of using this in AES is to transfer the data so that only the desired receiver with a specific key would be able to retrieve the original data. The fault detection is necessary for AES hardware implementation. This is because of following two reasons. Natural faults caused by defects in gates may cause erroneous output in encryption or decryption. Also the attackers can inject certain faults in the AES for retrieving the key. Due to this reasons, finding a suitable fault detection scheme has been an issue in AES. Not only this scheme should be able to detect a reasonable portion of faults but there should also be a compromise between the fault detection capability of the scheme and the overhead in terms of area and critical path delay.

The S-box and inverse S-box are preferred to be implemented by using the Look Up Tables [9-10].The look-up table based S-box (inverse S-box) is a table of 256 bytes which maps the bytes of the input state to new bytes using 256 × 8 memory cells. In the parity-based method, the output parity bits of each transformation in every round are predicted from the inputs. As mentioned in [9], for increasing the fault coverage and also to reduce the complexity of the predicted parities, parity predictions are done for each byte of the input state, i.e., 16 parity bits are used for fault detection of the output of each transformation in every round. The parity predictions of the S-box and inverse S-box in [10] are based on look-up table implementations in which 512×9 memory cells are used to generate the 8-bit output as well as the predicted parity bit which detects the input state errors. If a correct 9-bit address (one byte of the input state with its parity) is decoded, the corresponding output byte with its associated parity bit is produced as in figure 4. In this scheme, the parity of each block is predicted and it is then compared to the actual parity. The result of this comparison is the error indication flags of the corresponding block. When this error indicating flags are zero we assume that, no faults are present in the AES cipher.When the error indicating flags are

one, we assume that, faults are present AES cipher, which causes erroneous output.

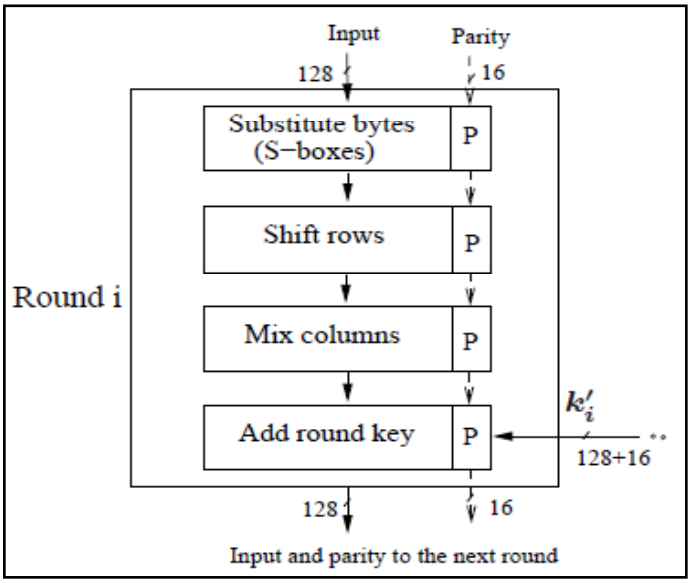

Fig. 4: Parity Based Fault Detection Structure of I[th] Round in Encryption

IV. Results

The Results of 128 bit AES is shown in this section. The design of 128 bit AES encryption, decryption and fault detection was done in VHDL Language and simulated using ModelSim 6.5. Encryption takes 128 bits as input and output with key length of 128 bits. After completing the 10 rounds, ciphertext is obtained as output. Also decryption takes 128 bits as input and output with key length of 128 bits. AES cipher involves both encryption and decryption. The mode is used for selecting the encryption and decryption operations. When mode = 1, encryption occurs and ciphertext is obtained as input. When mode = 0, decryption occurs and plaintext is obtained as output. Power Analysis has been done using PlanAhead tool in Xilinx 13.2. The power consumption of 128 bit AES is shown in fig. 5. The power consumption of 128 bit AES is 1342 mw.

The results of parity based fault detection schemes for Advanced Encryption Standard is shown below. When the error flags are zero, we assume that no faults are present in the AES cipher as in figure 6. Also when the error flags are one, we assume that faults are present in the AES cipher as in fig. 7.

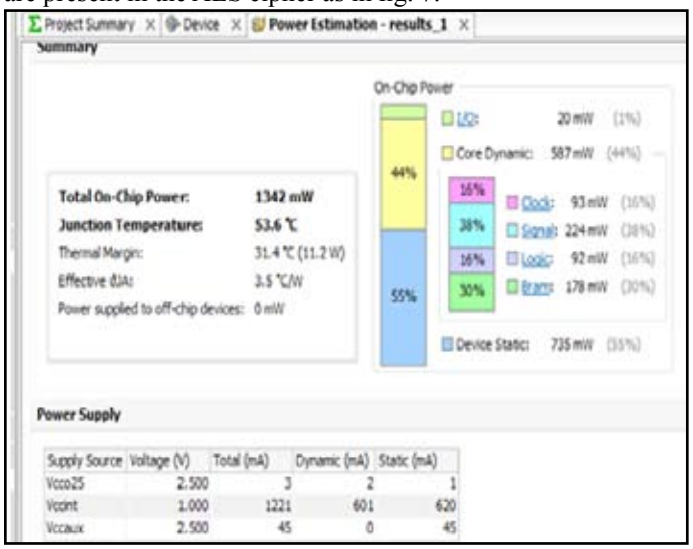

Fig. 5: Power Consumption in 128 Bit AES Cipher

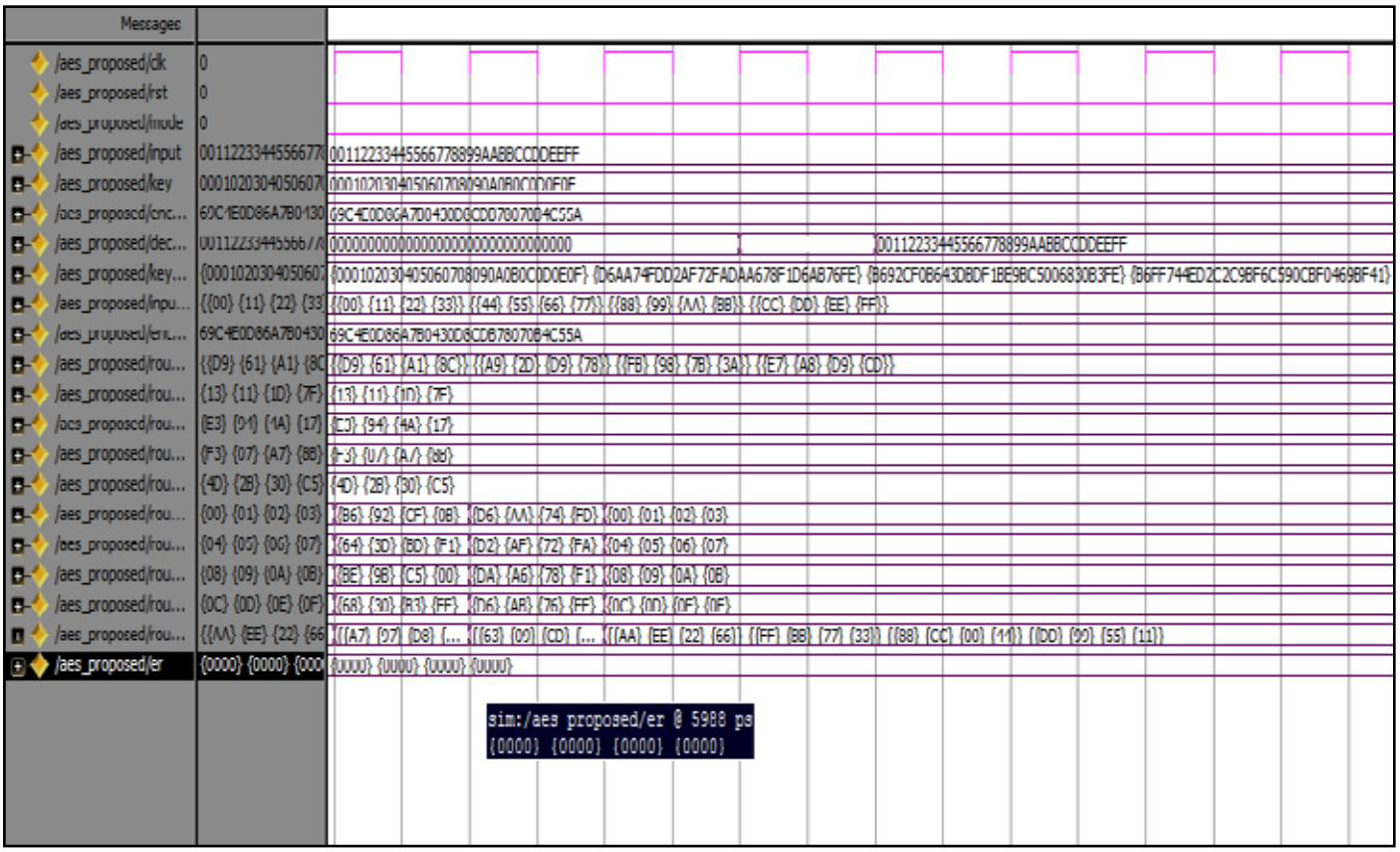

Fig. 6: Fault Free Waveform for AES

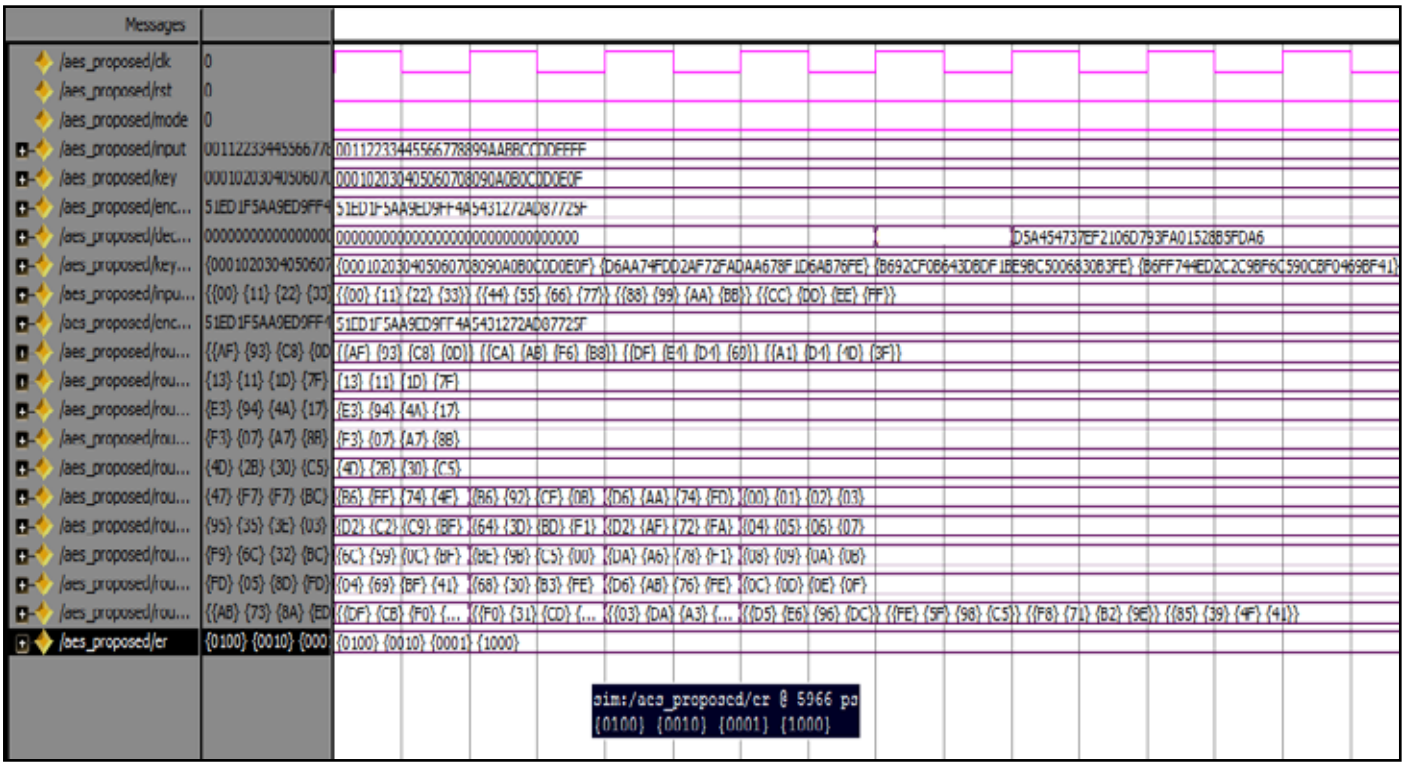

Fig. 7: Faulty Waveform for AES

V. Conclusion

The 128 bit Advanced Encryption Standard (AES) was designed using Xilinx tool. In this design, S-Box and inverse S-Box is implemented by using Look Up Table approach. Parity based fault detection scheme for this S-Box and inverse S-Box is presented in order to find the faults in the hardware implementation for high performance AES implementations.

VI. Acknowledgment

The authors would like to thank the reviewers for discussions and valuable suggestion.

References

[1] Nat. Inst. Standards Technol. (NIST),"Federal Information Processing Standards (FIPS) Publication 197", Advanced Encryption Standard,Nov 2001.

[2] Avi kak AES: The Advanced Encryption Standard Lecture Notes on,"Computer and Network Security", April 16, 2012.

[3] T. Good, M. Benaissa,"Very small FPGA application-specific Instruction processor for AES", IEEE Trans. Circuits Syst. I, Reg. Papers, Vol. 53, No. 7, pp. 1477–1486, Jul. 2006.

[4] H. Yen, B. F. Wu,"Simple error detection methods for hardware implementation of advanced encryption standard", IEEE Trans. Computers, Vol. 55, No. 6, pp. 720-731, June 2006.

[5] A.E. Cohen,"Architectures for Cryptography Accelerators", Ph.D dissertation,Univ.Minnesota, Twin Cities, Sep. 2007.

[6] R. Karri, K. Wu, P. Mishra, Y. Kim,"Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric cryptanalysis of symmetric block ciphers", IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., Vol. 21, No. 12, pp. 1509–1517, Dec. 2002.

[7] R. Karri, K. Wu, P. Mishra, K. Yongkook,"Fault-based side-channel cryptanalysis tolerant Rijndael symmetric block cipher architecture", in Proc. DFT, Oct. 2001, pp. 418–426.

[8] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, V. Piuri,"Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard", IEEE Trans. on Computers, Vol. 52, No. 4, pp. 492-505, April 2003.

[9] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri,"An efficient hardware-based fault diagnosis scheme for AES: performances and cost", In Proc. of the IEEE International Symposium on Defect and Fault Tolerance in VLSI System (DFT2004), Cannes, France, pp. 130-138, Oct. 2004.

[10] K. Wu, R. Karri, G. Kuznetsov, M. Goessel,"Low cost concurrent error detection for the advanced encryption standard", In Proc. of the 2004 International Test Conference, Charlotte, NC, USA, pp. 1242-1248, October 2004.

Jishamol T.K received her B.Tech Degree in Electronics and Communication Engineering from College of Engineering Thalassery, Cochin University, Kerala in 2010, the M.Tech Degree in VLSI Design from Karunya University, Karunya Nagar, Tamilnadu in 2013.

Ginu Thomas received her B.Tech Degree in Electronics and Communication from Oxford engineering college, Anna University, Tamilnadu in 2011, the M.Tech Degree in VLSI Design from Karunya University, Karunya nagar, Tamilnadu in 2013.