Fault Detection Scheme for AES Using Optimization for Mix Column

H.K.Reshma Ramesh[1], **Dr. N. Nagarajan**[2]
[1]**Department of Electronics and Communication Engineering, Anna University**
Coimbatore,Tamil Nadu, India

[2] **Department of Computer science and Engineering, CIET, Anna University**
Coimbatore,Tamil Nadu, India

Abstract
In this paper for existing concurrent structure independent fault detection schemes with new technique for the fault detection of sub bytes and inverse sub bytes using the relation between the input and output of the S-box, the formulation of mix column are implemented for AES, which results in the reduction of area coverage and power consumption along with the error coverage of greater than 99% of the existing scheme. The proposed scheme and existing scheme have been implemented on the most recent Xilinx Sparton FPGA, their area and power requirements are compared and it is proved that proposed technique makes the fault detection for AES efficient in terms of area coverage(in terms of gate count) and power consumption.

Keywords: *AES(Advanced encryption standard), FPGA(Field programmable gate array), Galois field, Mix column*

1. Introduction

NIST (National Institute of Standards and technology) in 2001, accepts the advanced encryption standard (AES) as the symmetric cryptography standard. That is intended to replace DES and its new version 3DES which has two attractions. First, with its 168 bit key length, it overcomes the vulnerability to brute force attach. Second, the encryption algorithm in 3DES is same as in DES, along with the principle drawback of being relatively sluggish in software. The original DES was designed for mid 1970s hardware implementation and could not produce efficient software code. 3DES, which has 3 times as many round as DES, is relatively slower. A secondary drawback is that both DES and 3DES use a 64 bit block size. For reason of both efficiency and security, a larger block size is desirable. NIST specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key length of 128, 192 and 256 bits. As described in [1], after the initial evaluation of AES for different categories of criteria such as security, cost, algorithm and implementation characteristics and the final evaluation with the criteria like general security, software implementations, restricted space environment, hardware implementations, attacks on implementation, encryption V/S decryption, key agility, other versatility and flexibility and potential for instruction level parallelism NIST selected Rijndale as the proposed AES algorithm. Rijndale was designed to have 3 important characteristics. First, resistant against all known attacks. Second, speed and code compactness on a wide range of platforms. Third, design simplicity. The motive force behind the wide spreaded usage of AES in different critical applications is the hardware and software integrated implementation which results in high speed and cost effective application with high level security. The reliable architecture [2] of AES has been implemented for bit stream security mechanism in the FPGAs for increasing the reliability of the FPGA based design [3] used in the recent Xilinx virtex FPGA family [4]. The hardware and software integrated implementation yields the high speed and cost effective applications with high level of security. However, the natural and malicious injected faults reduces its reliability and may results in the leakage of confidential information leakage.

2. Related works

Fault detection and possibly fault tolerance are undoubtedly key issues when designing a crypto processor custom VLSI architecture for implementing the AES cryptosystem. Since it is considerably more complex than the DES cryptosystem it replaces. In fact, AES executes a very nonlinear algorithm and has an iterative structure requiring several repetitions of the same basic pattern of operations [2]. Therefore an AES crypto processor is larger, more complex, and hence more likely to be subject to faults. Moreover, fault detection is a desirable property for preventing malicious attacks, aimed at extracting sensitive information like the secrete key from the device. Attacks against the cryptographic hardware can be divided into two classes: invasive and noninvasive. Invasive attacks are based on reverse engineering and require special laboratory equipments, making it too expensive. Non invasive attacks, also called side channel attacks, exploit hardware implementation weaknesses. One possible form of attack is to observe characteristics of the hardware during execution, such as power consumption, execution time or electromagnetic emissions to extract statistical information that can eventually lead to the recovery of secret key. For example, Differential power analysis (DPA) exploits the power consumption of the device. A second form of attack is called differential fault analysis (DFA) [6] or simply fault attack which is based on the injection of a transient fault in the cryptosystem core. The attacker can use several techniques to inject transient faults in the cryptographic hardware. For example, Glitch attacks consists on submitting VCC (power), GND(ground) or the clock inputs to stress conditions. To do this, peaks on the power supply voltage or an irregular clock can be used. Another technique is the light attack; intensive light (such as laser) may cause disturbances on semiconductors resulting in transient faults.

Ciphered communication is very sensitive to errors in the input data or faults occurring during the computation due to the strong non linearity of the encryption function. The analysis of the effect of fault occurring [5] during encryption process for AES has shown that even a single bit error leads after a few rounds of the algorithm to a completely corrupted result. Here a parity based error detection code was proposed for the AES cipher using one parity bit for each byte of the 128 bit long input. The redundancy level was determined based on the round operation, which also shows how parity code can be propagated through all of the round operation activating impressive fault coverage (fault detection probability). All the odd order faults are detected. Over all fault coverage is over 99%.

The DFA technique is explained in paper [6] [7] &[8]. In september 1996, Bonch, Demillo and Lipton from Bellore announced a new type of cryptanalytic attack which exploits computational errors to find cryptographic keys. Their attack is applicable to public key cryptosystem such as RAS, excluding secret algorithm. In [6] P.Dusart, G. Letournex, O.vivolo explains how a differential fault analysis(DFA) works on AES 128, 192 or 256 bits, which shows that concurrent checking for cryptographic chips has also a great potential for detecting (deliberate fault injection attacks) where faults are injected into a cryptographic chip to break the key. Dusart et al also broke the 128 bit AES under the assumption that you can physically modify the hardware of AES device. This attack requires 34 pairs of differential inputs and outputs to obtain the final round key.

Piret and Quisquater broke AES with two erroneous cipher text [7] under the assumption that the error occurs between the antepenultimate and the penultimate mix columns. Fault attacks consists of forcing a cryptographic device to perform some erroneous operation hoping that the result of that wrong behavior will leak information about the secrete parameter involved. These techniques have been increasing, studied and the content of public key cryptosystems and its extension to the private key setting were studied [7]. Here a number of possible attacks to AES algorithms are presented. In addition, fault injection tools considering permanent and transient fault have been largely studied in the last few years and can be reused and / or adapted for this analysis. The one step ahead of previous works, the combine concepts of collision and fault attacks [8], i.e. by inducing faults to create collisions which need only a moderate number of faults to get the key.

It becomes obvious how many secure IP has to be protected against both natural and intension faults. Counter measures against fault attacks can be deployed in hardware or software and generally help circuits to avoid, detect and / or correct faults [9], [10], [11], [12], [13], [14], [15] and [16]. Certain active protections use sensors and detectors to infer abnormal circuit behavior. Passive protections such as randomization of the clock cycles or bus and memory encryption may also be used to increase the difficult of successfully attacking a device. However in practice most proposed schemes are based on classic error detecting techniques using space and time redundancies [9], [10]. [13], [14] describes the two counter measures to protect cryptographic implementation against fault injection based on each of the above said techniques. But these counter measures are proved to be costly in terms of throughput or circuit idea. The better architectures [10] also double data rate techniques can be used to improve the throughput [11] compared to [9] for protecting hardware implementation of AES. First one based on portioning, is an efficient and effective method with an assumption that probabilistic attacks have a high wire distortion rate. The second architecture based on robust code for which on assumption can be made above the wire distortion rate. The later is proved the good resistance against single and multiple fault errors. In the former the AES Rijndael is divided into two blocks: linear and nonlinear, where the nonlinear block consists the multiplicative inverse [1] of the Rijndael S-box. In order to reduce the area overheads it is proposed to check only a few bits (2 bits) of the result. Then the linear part, every column of AES is associated with an 8 bit parity, namely the X-OR between the 4 bytes of the column. It yields a 32 bit redundancy for the complete algorithm which is computed independently. The nonlinear part allows good error detection for faults with high multiplicities.

Carlos R. Moratelli, Erika Cota and Marcelo S. Lubaszewski proposed a cost effective solution [12] to protect cryptographic cores against malicious attacks whish has the main advantages of being independent of the core implementation. In 2003, Karri. Et. Al. proposed parity based detection technique for general SP block ciphers [13] where single bit errors are easily detected by these cheaper codes. However the size of the table required by the substitution box is enlarged. In addition, the paper did not address the error techniques for some specific functions such as mix columns in AES. He applied the structure of [9] to AES and used one bit parity for a 128 bit data block. This counter measures for AES have lower coverage or higher overhead due to the use of more complex codes. As a prevention to AES from suffering from differential fault analysis (DFA), [15] proposed a simple symmetric and high fault coverage error detection scheme. Here linear behavior of each operation in AES is used to design a detection scheme. This scheme only uses an (n+1, n) CRC to detect the errors, where n $\in$ (4, 8, 16) and the parity of the output of each operation is predicted in a simple fashion.

The counter measures for AES [13] have lower performance and lower coverage or higher overhead due to the use of more complex codes. We believe that error detecting codes can provide a useful protection against fault attacks and in general, against errors occurring during the encryption process. They can often provide full coverage of single bit errors and high coverage of multiple bit errors. The actual coverage depends on many configurable parameters such as redundancy, granularity and validation frequency; although duplication can provide better coverage figures, it has base overhead that is much larger than that of EDCs. EDCs can protect against single bit errors occurring in the data path, which are most likely being faults and when injected maliciously, are the most dangerous fault attacks. More over EDCs can provide high coverage for multiple bit errors, which are the most common in fault attacks. In this case, the coverage depends heavily on the fault pattern and on the redundancy. The operation centered approach in [16], first enumerated the arithmetic and logic operation included in the cipher, later the efficiency and the hardware complexity of several error detecting codes for each operation is analyzed. This technique supports negligible performance degradation with a reasonable area overhead in AES architecture.

3. AES Basics

The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192 or 256 bits. In both encryption and decryption process four different stages are used, one of permutation and three of substitution. They are, substitute bytes, shift rows, mix column and add round key.

Substitute byte transformation which is also called as sub bytes is a simple table lookup. AES defines a 16X16 matrix of bytes values called an S- box that contain a permutation of all possible 256 eight bit value. Each individual byte of state is mapped into a new byte in the following way: the left most four bits of the byte are used as a row value and the right most four bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8 bit output. An S-box is the multiplicative inverse of a Galois field GF (2^8) followed by an affine transformation. In the decryption process the affine transformation is executed prior to the inversion [17]. The irreducible polynomial used by a Rijndeal S-box is $m(x)=x^8+x^4+x^3+x+1$.

Shift rows: output of the sub byte process in the form of state which is a matrix of 4X4, will be the input to shift rows process. The first row of the state is not altered. For the second, third and fourth row the circular shift of one byte, two byte and three byte left shift will be performed respectively. The inverse shift row transformation called Inv shift rows performs the circular shift in opposite direction for each of the last three rows.

Mix column: each column of the input state can be considered as a polynomial of degree three, and respective output state column values will be sum of products of polynomial and the coefficients, where the coefficients are the respective row elements in the constant mix column matrix.

The polynomial for forward mix column is

$$C(x) = \{01\}\, x^3 + \{02\}\, x^2 + \{03\}\, x + \{01\} \qquad (a)$$

The polynomial for inverse mix column is

$$C^{-1}(x) = \{09\}\, x^3 + \{0E\}\, x^2 + \{0B\}\, x + \{0D\} \qquad (b)$$

Add round key: here bit by bit X-OR will be performed between 128 bit of state with 128 bit key.

Key expansion: the key expander generator [17] generates 11 sets of 128 bit round key from one 128 bit secret key by using a four byte S-box. These round keys can be prepared on the fly in parallel with the encryption process. In the decryption process these sets of keys are used in reverse order.

4. Existing Error detection scheme for AES encryption

The error detection scheme can be incorporated in every round of AES encryption and decryption which consists of all the 4 basic formulations [18]. In the existing error detection scheme, error detection method is implemented as a combination of two techniques which uses the input and output states of two basic formulations i.e. first two and next two out of 4 basic steps of AES. The fault detection scheme is shown in figure 1.

Technique one: This fault detection technique make use of input and output state of combination of Sub Bytes and shift rows formulations of basic AES. Here $e_{r,c}$ is the error indication flag for the S-box with the input and output of $S_{r,c}$ and $S'_{r,c}$ respectively, where output state flags can be written as 16 formulations as follows:

$$e_{r,c} = P_{(M_{r,c^*} S'_{r,c^*} + m_{r,c^*})} + u'_{r,c^*}, \quad 0 \leq r, c \leq 3, \tag{1}$$

where, $c^* = (r+c)\bmod 4$ (2)

$$u'_{r,c^*} = Ms' + m = u' \tag{3}$$

$$s' = [s'_0, s'_1, s'_2, s'_3, s'_4, s'_5, s'_6, s'_7]^T \tag{4}$$

$$m = [s_{6,0}, s_{7,6,1}, s_{7,2,0}, s_{6,3,1}, s_{7,6,4,2}, s_{7,5,3}, s_{6,4}, s_{7,5}]^T \tag{5}$$

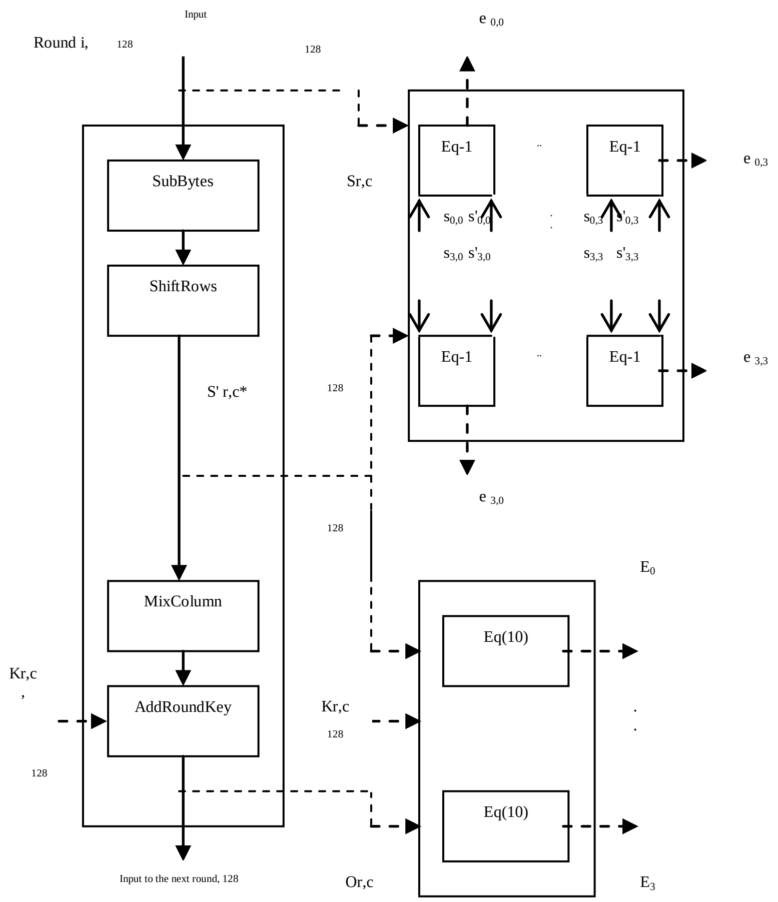

Figure: 1

$$u' = [u', 0, 0, 0, 0, 0, 0, 0]^t \tag{6}$$

$$u' = (s_0 \vee s_1 \vee s_2 \vee s_3 \vee s_4 \vee s_5 \vee s_6 \vee s_7) \vee (\overline{s'_0} \vee \overline{s'_1} \vee s'_2 \vee s'_3 \vee s'_4 \vee \overline{s'_5} \vee \overline{s'_6} \vee s'_7) \tag{7}$$

$$M = \begin{pmatrix} s_{6,5,2} & s_{5,4,1} & s_{7,5,3,0} & s_{6,4,2} \\ s_{7,5,3,2,0} & s_{6,4,2,1} & s_{7,6,5,4,3,1} & s_{7,6,5,4,3,2,0} \\ s_{6,4,3,1} & s_{7,5,3,2,0} & s_{7,6,5,4,2} & s_{7,6,5,4,3,1} \\ s_{7,6,4,0} & s_{6,5,3} & s_{6,0} & s_{7,5} \\ s_{7,6,2,1} & s_{7,6,5,1,0} & s_{5,3,1} & s_{4,2,0} \\ s_{7,3,2} & s_{7,6,2,1} & s_{6,4,2,0} & s_{5,3,1} \\ s_{4,3,0} & s_{7,3,2} & s_{7,5,3,1} & s_{6,4,2,0} \\ s_{5,4,1} & s_{4,3,0} & s_{6,4,2} & s_{7,5,3,1} \\ s_{7,5,3,1} & s_{7,6,5,2,0} & s_{7,6,5,4,1,} & s_{7,6,3,0} \\ s_{7,6,5,4,3,2,1} & s_{5,3,2,1} & s_{4,2,1,0} & s_{6,4,3,1} \\ s_{7,6,5,4,3,2,0} & s_{6,4,3,2} & s_{5,3,2,1} & s_{7,5,4,2,0} \\ s_{6,4} & s_{6,4,3,2,0} & s_{7,5,3,2,1} & s_{7,5,1} \\ s_{3,1} & s_{6,4,3,2,1} & s_{7,5,3,2,1,0} & s_{7,3,2} \\ s_{4,2,0} & s_{7,5,4,3,2} & s_{6,4,3,2,1} & s_{4,3,0} \\ s_{5,3,1} & s_{6,5,4,3,0} & s_{7,5,4,3,2} & s_{5,4,1} \\ s_{6,4,2,0} & s_{7,6,5,4,1} & s_{6,5,4,3,0} & s_{6,5,2} \end{pmatrix} \tag{8}$$

Parity of Ms'+m which can be expressed as P(Ms'+m) [18]can be computed as follows:

$$P(Ms'+m) = s_0(s'_b+s'_c) + s_1 s'_b + s_2 s'_d + s_3 s'_4 + s_4(s'_c+s'_3) + s_5 s'_a + s_6(s'_d+\overline{s'_6}) + s_7\,\overline{(s'_5+s'_4)} = u' \tag{9}$$

where $s'_a = s'_0 + s'_2 + s'_3 + s'_5$, $s'_b = s'_a + s'_7$, $s'_c = s'_1 + s'_4 + s'_6$ and $s'_d = s'_2 + s'_7$.

According to Eq (1), 16 error indication flags for the sub bytes and shift rows transformation, i.e. one error indication flag for each byte are obtained. Let us consider Eq (3) for the input $s=0=(0,0,\ldots,0) \in GF(2^8)$. For this input, the correct output is $s' = \{63\}_h = (0,1,1,0,0,0,1,1) \in GF(2^8)$. If the erroneous output is not $s' = \{63\}_h = (0,1,1,0,0,0,1,1) \in GF(2^8)$, in the right hand side of Eq (3), we have u' = 1, where the left-hand is zero. Therefore, the erroneous output is detected.

Technique two: These fault detection techniques maks use of input and output state of combination of mix column and add round key formulations of basic AES. Here input state of mix column is referred as s' and output state of add round key formulation is referred as o and the key used in the add round key will be k. considering the state in terms of rows and column these parameters can be represented as $[S'_{r,c}]^3_{r,c=0}$, $k = [k_{r,c}]^3_{r,c=0}$ and $o = [o_{r,c}]^3_{r,c=0}$. all the three parameters can be formulated as follows:

$$\sum_{r=0}^{3} (S'_{r,c^*} + k_{r,c} + o_{r,c}) = 0 \in GF\left(2^8\right), 0 \leq c \leq 3 \tag{10}$$

where $c^* = (r+c)\bmod 4$ and each summation is over $GF(2^8)$ which consists of eight modulo-2 additions.

The 8 bit error indication flag is introduced as;

$$E_c = \sum_{r=0}^{3} (S'_{r,c*} + k_{r,c} + o_{r,c}) = 0 \in GF\left(2^8\right), 0 \le c \le 3 \quad (11)$$

if all the 32 bits of such flag in above expression is zero i.e. $E_c = 0 = (0, 0, \ldots, 0) \in GF(2^8)$, $0 \le c \le 3$, then the encryption process is error free.
For AES decryption, as the AES decryption also consists of four transformations i.e. inv shifts rows, inv sub bytes, add round key and inv mix column, for error detection the same techniques are used as the AES encryption [18].

5. Optimization technique for AES mix column

Although AES is used in many different applications, hardware implementations of the algorithm focus mostly on throughput optimization. In the existing method of error detection, though the error detection is efficient which provides 99% to 100% of error coverage on the maliciously injected faults, since no optimization technique is followed while implementing the 4 different basic formulation of AES i.e. shift rows, sub bytes, add round key, mix column which results in more power requirement and area covered (in terms of gate count) to implement the entire encryption and decryption along with error detection technique for the hardware implementation. In this paper we have used the optimized light weight technique [19] for the implementation of mix column which is one of the AES basic formulations, along with the existing error detection technique. By using new formulation for the mix column we proved considerable changes in the power requirement and area coverage.
Optimization for mix column: In AES encryption process the forward mix column transformation is called as mix column. The mix column process operates on each column of the input states separately. Every byte of each column is replaced with a new value which is going to be computed using all four bytes in that column, such transformation is defined by the following matrix multiplication with the state.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} = \begin{pmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{pmatrix} \quad (12)$$

The each byte in the output sate product matrix is the sum of product of elements of corresponding row and column. Here the individual additions and multiplications are performed in $GF(2^8)$. The single column transformation in the mix column j $(0 \le j \le 3)$ of the state can be expressed as ;

$$\begin{aligned} s'_{0,j} &= (2*s_{0,j}) \oplus (3*s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2*s_{1,j}) \oplus (3*s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2*s_{2,j}) \oplus (3*s_{3,j}) \\ s'_{3,j} &= (3*s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2*s_{3,j}) \end{aligned} \quad (13)$$

As per the above expressions the mix column transformation requires only multiplication by {02} and {03}, which generally implemented using simple shifts, conditional XORs and XORs. In more efficient way the mix column transformation can be implemented which eliminates the shifts and conditional XORs. The above equation set which represents the mix column transformation on a single column [19] can be rewritten using identity {03}x = [{02}x]x as follows :

$$\begin{aligned} Temp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\ s'_{0,j} &= s_{0,j} \oplus Temp \oplus [2*(s_{0,j} \oplus s_{1,j})] \\ s'_{1,j} &= s_{1,j} \oplus Temp \oplus [2*(s_{1,j} \oplus s_{2,j})] \\ s'_{2,j} &= s_{2,j} \oplus Temp \oplus [2*(s_{2,j} \oplus s_{3,j})] \\ s'_{3,j} &= s_{3,j} \oplus Temp \oplus [2*(s_{3,j} \oplus s_{0,j})] \end{aligned} \quad (14)$$

The implementation of the optimized mix column transformation as per the above expressions has been done [19] using 116 XOR with 2 input or 52 XOR with 2 inputs + 32 XOR with 3 inputs, with total 84 XORs.

In AES decryption process, the inverse mix column transformation is called Inv mix columns. This transformation is defined by the following matrix multiplication.

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} = \begin{pmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{pmatrix} \quad (15)$$

The each byte in the output state product matrix is the sum of product of elements of corresponding row and column. Here the individual additions and multiplications are performed in $GF(2^8)$. The single column transformation in the mix column j $(0 \le j \le 3)$ of the state can be expressed as ;

$$\begin{aligned} s'_{0,j} &= (0E*s_{0,j}) \oplus (0B*s_{1,j}) \oplus (0D*s_{2,j}) \oplus (09*s_{3,j}) \\ s'_{1,j} &= (09*s_{0,j}) \oplus (0E*s_{1,j}) \oplus (0B*s_{2,j}) \oplus (0D*s_{3,j}) \\ s'_{2,j} &= (0D*s_{0,j}) \oplus (09*s_{1,j}) \oplus (0E*s_{2,j}) \oplus (0B*s_{3,j}) \\ s'_{3,j} &= (0B*s_{0,j}) \oplus (0D*s_{1,j}) \oplus (09*s_{2,j}) \oplus (0E*s_{3,j}). \end{aligned} \quad (16)$$

The above equation set, to simplify its hardware can be implemented as follows:

$$\mathrm{Temp} = 09 * (s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j})$$
$$s'_{0,j} = s_{0,j} \oplus \mathrm{Temp} \oplus 2*[2*(s_{0,j} \oplus s_{2,j})] \oplus 2*[(s_{0,j} \oplus s_{1,j})]$$
$$s'_{1,j} = s_{1,j} \oplus \mathrm{Temp} \oplus 2*[2*(s_{1,j} \oplus s_{3,j})] \oplus 2*[(s_{1,j} \oplus s_{2,j})]$$
$$s'_{2,j} = s_{2,j} \oplus \mathrm{Temp} \oplus 2*[2*(s_{0,j} \oplus s_{2,j})] \oplus 2*[(s_{2,j} \oplus s_{3,j})]$$
$$s'_{3,j} = s_{3,j} \oplus \mathrm{Temp} \oplus 2*[2*(s_{1,j} \oplus s_{3,j})] \oplus 2*[(s_{3,j} \oplus s_{0,j})] \quad (17)$$

The implementation of the optimized Inv mix column transformation as per the above expressions has been done [19] using 198 XOR with 2 input. Implementing optimized forward and inverse mix column transformation uses 116 + 198 XOR gates. The considered optimized mix column design for AES hardware implementation have less complexity in terms of gate size and number of clock cycles compared to other relevant optimization techniques [20] [21] [22].

6. AES FPGA implementation and comparison

The existing structure independent error detection scheme for AES along with optimization technique for mix column used LUT based S-boxes and inverse S-boxes. In this paper we have implemented both the structure independent error detection scheme for AES with and with out optimization techniques. For FPGA implementation, we have used VHDL as the design entry using modelsim 6.3. further more, the synthesis is performed using Xilinx 8.1 on sperton 3E family. We have implemented the original AES with optimization technique for mix column using LUT based S-boxes and inverse S-boxes on sparton 3E (XC3S100E) device. Using Xilinx 8.1 synthesis tool the area coverage and power requirement for substitute byte transformation with and with out look up table and mix column with and without optimization technique in both encryption and decryption process is compared. And it is proved that, using LUTs for S-boxes and optimized technique for mix column the considerable saving in terms of area coverage (in terms of gate count) and power requirements. The result of entire implementation and comparisons is presented in the tables below:

Table 1: AES Encryption

Transfor mation	Area Covered (In terms of gate count)		Power Requirement (mW)	
Sub Bytes	Without LUTs	With LUTs	Without LUTs	With LUTs
	1,348	1,264	45	39
Mix Column	Without optimization	With optimization	Without optimization	With optimization
	23,540	640	80	58

Table 2: AES Decryption

Transfor mation	Area Covered (In terms of gate count)		Power Requirement (mW)	
Sub Bytes	Without LUTs	With LUTs	Without LUTs	With LUTs
	1,576	1,258	56	39
Mix Column	Without optimization	With optimization	Without optimization	With optimization
	23,726	1,294	89	66

7. Conclusion

In this paper, we have studied a number of fault detection schemes and optimization techniques for the AES encryption and decryption. The existing structure independent error detection scheme for AES along with optimization technique for mix column used LUT based S-boxes and inverse S-boxes. In this paper we have implemented both the structure independent error detection scheme for AES with and with out optimization techniques. Our FPGA implementation showed the considerable reduction in the area coverage and power requirement for substitute byte transformation with look up table and mix column with optimization technique in both encryption and decryption.

Acknowledgments

The authors would like to thank the reviewers for their comments. Thanks to my guide Dr. N. Nagarajan for his guidance and support for the work.

References

[1] "Cryptography and network security" – Principles and practices, William Stallings.
[2] National Institute of Standards and Technologies, "Announcing the Advanced Encryption Standard (AES),"Federal Information Processing Standards Publication, no. 197, Nov. 2001.
[3] S. Trimberger, "Security in SRAM FPGAs," IEEE Design and Test of Computers, vol. 24, no. 6, p. 581, Nov. 2007.
[4] Xilinx, http://www.xilinx.com/, 2010.
[5] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error Analysis and Detection Procedures for a Hardware

Implementation of the Advanced Encryption Standard," IEEE Trans. Computers,vol. 52, no. 4, pp. 492-505, Apr. 2003.

[6] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on AES," Proc. Int'l Conf. Applied Cryptography and Network Security (ACNS '03), pp. 293-306, Oct. 2003.

[7] G. Piret and J.J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad," Proc. Int'l Workshop Cryptographic Hardware and Embedded Systems (CHES '03), pp. 77-88, Sept. 2003.

[8] J. Blomer and V. Krummel, "Fault Based Collision Attacks on AES," Proc. Int'l Workshop Fault Diagnosis and Tolerance in Cryptography (FDTC '06), pp. 106-120, Oct. 2006.

[9] T.G. Malkin, F.X. Standaert, and M. Yung, "A Comparative Cost/ Security Analysis of Fault Attack Countermeasures," Proc. Int'l Workshop Fault Diagnosis and Tolerance in Cryptography (FDTC '06), pp. 159-172, Oct. 2006.

[10] M. Karpovsky, K.J. Kulikowski, and A. Taubin, "Differential Fault Analysis Attack Resistant Architectures for the Advanced Encryption Standard," Proc. Conf. Smart Card Research and Advanced Applications (CARDIS '04), vol. 153, pp. 177-192, Aug. 2004.

[11] P. Maistri and R. Leveugle, "Double-Data-Rate Computation as a Countermeasure against Fault Analysis," IEEE Trans. Computers vol. 57, no. 11, pp. 1528-1539, Nov. 2008.

[12] C. Moratelli, E. Cota, and M. Lubaszewski, "A Cryptography Core Tolerant to DFA Fault Attacks," Proc. Ann. Symp. Integrated Circuits and Systems Design (SBCCI '06), pp. 190-195, Sept. 2006.

[13]R. Karri, G. Kuznetsov, and M. Goessel, "Parity-Based Concurrent Error Detection of Substitution-Permutation Network Block Ciphers," Proc. Int'l Workshop Cryptographic Hardware and Embedded Systems (CHES '03), pp. 113-124, Sept. 2003.

[14] K. Wu, R. Karri, G. Kuznetsor, and M. Goessel, "Low Cost Concurrent Error Detection for the Advanced Encryption Standard", Proc. Int'1 Test Conf. '04, pp. 1242-1248,Oct.2004.

[15] C.H. Yen and 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.

[16] L. Breveglieri, I. Koren, and P. Maistri, "An Operation-Centered Approach to Fault Detection in Symmetric Cryptography Ciphers," IEEE Trans. Computers, vol. 56, no. 5, pp. 534-540, May 2007.

[17] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," Proc.Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '01), pp. 239-254, Dec. 2001.

[18]Mehran Mozaffari-Kermani, Arash Reyhani-Masoleh, "Concurrent Structure-Independent Fault Detection Schemes for the Advanced Encryption Standard "IEEE TRANSACTIONS ON COMPUTERS, VOL. 59, NO. 5, pp.608-622, MAY 2010.

[19] Eslam Gamal Ahmed, Eman Shaaban, Mohamed Hashem, "Lightweight Mix Columns Implementation for AES" Proceedings of the 9th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC '09), pp. 253-258

[20] Hua Li and Zac Friggstad "An Efficient Architecture for the AES Mix columns Operation" Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on Volume, Issue, 23-26 May 2005 Page(s): 4637-4640 Vol. 5

[21] H. Kuo, I. Verbauwhede, and P. Schaumont, "A 2.29 gbits/sec, 56 mw non-pipelined rijndael aes encryption ic in a 1.8v, 0.18 um cmos technology." [Online]. Available citeseer.nj.nec.com/kuo02gbitssec.html

[22] S. Mangard, M. Aigner, and S. Moninikus, "A highly regular and scalable aes hardware architecture," *IEEE Transactions on Computers*, April 2003, vol. 52, no. 4, pp. 483–491.

First Author H.K. Reshma Ramesh received the BE degree in electronics and communication engineering from Kuvempue university. She has total 9 years of teaching experience .She is currently working towards the M.Tech (By Research) degree at the department of Electronics and communication engineering, Anna University, Coimbatore. Her research interest includes secure cryptographic systems, Embedded applications and VLSI reliability.

Second Author Nagarajan.N received his B.Tech and M.E. degree in Electronics Engineering at M.I.T Chennai. He received his Ph.D in faculty of I.C.E. from Anna University, Chennai. He is currently working as Dean, CSE, C.I.E.T, Coimbatore. His specialization includes optical, wireless Adhoc and sensor networks.