J Electron Test (2009) 25:225–245
DOI 10.1007/s10836-009-5108-4

Fault Detection Structures of the S-boxes and the Inverse S-boxes for the Advanced Encryption Standard

Mehran Mozaffari-Kermani · Arash Reyhani-Masoleh

Received: 28 April 2008 / Accepted: 1 July 2009 / Published online: 17 July 2009

Abstract Fault detection schemes for the Advanced Encryption Standard are aimed at detecting the internal and malicious faults in its hardware implementations. In this paper, we present fault detection structures of the S-boxes and the inverse S-boxes for designing high performance architectures of the Advanced Encryption Standard. We avoid utilizing the look-up tables for implementing the S-boxes and the inverse S-boxes and their parity predictions. Instead, logic gate implementations based on composite fields are used. We modify these structures and suggest new fault detection schemes for the S-boxes and the inverse S-boxes. Using the closed formulations for the predicted parity bits, the proposed fault detection structures of the S-boxes and the inverse S-boxes are simulated and it is shown that the proposed schemes detect all single faults and almost all random multiple faults. We have also synthesized the modified S-boxes, inverse S-boxes, mixed S-box/inverse S-box structures, and the whole AES encryption using the 0.18μ CMOS technology and have obtained the area, delay, and power consumption overheads for their fault detection schemes. Furthermore, the fault coverage and the overheads in terms of the space complexity and time delay are compared to those of the previously reported ones.

Responsible Editor: M. Goessel

M. Mozaffari-Kermani (✉) · A. Reyhani-Masoleh
Department of Electrical and Computer Engineering,
The University of Western Ontario,
London, Ontario, Canada
e-mail: mmozaff@uwo.ca

A. Reyhani-Masoleh
e-mail: areyhani@uwo.ca

Keywords Advanced encryption standard · Fault detection structures · Parity prediction · S-box · Inverse S-box

1 Introduction

The Advanced Encryption Standard (AES) is recently approved by NIST (National Institute of Standards and Technology) [10] as a replacement for the previous standards because of its good characteristics in terms of security, cost, and efficient implementations [10]. In encryption, the AES accepts a 128-bit plain text input. The key can be specified to be 128 (AES-128), 192 or 256 bits. In the AES-128, the cipher text is generated after ten rounds, where, each round consists of four transformations except for the final round which has three transformations. The decryption algorithm transforms the cipher text to the original plain text using the reverse procedure [10].

Each transformation in every round of encryption/decryption acts on its 128-bit input which is considered as a four by four matrix, called *state*, whose entries are eight bits. The transformations in each round of encryption except for the last round are as follows:

- *SubBytes*: The first transformation in each round is the bytes substitution, called *SubBytes*, which is implemented by 16 S-boxes. These S-boxes are nonlinear transformations which substitute the 128-bit input state with a 128-bit output state.
- *ShiftRows*: *ShiftRows* is the second transformation in which the four bytes of the rows of the input state are cyclically shifted to the left. The number of left shifts for each row is equal to the number

of that row. Let us denote rows as row_i where, i, $0 \leq i \leq 3$, is the row number. Then, for row_i, i shifts are required.
- *MixColumns*: The third transformation is *MixColumns* in which each entry in the output state is constructed by the multiplication of a column in the input state with a fixed polynomial over $GF(2^8)$.
- *AddRoundKey*: The final transformation is *AddRoundKey* in which a 128-bit roundkey is added modulo-2 to the input state.

Fault detection is an essential part of the AES hardware implementation. This is mainly because natural faults may cause erroneous output in the encryption/decryption and more importantly an attacker may inject faults during the AES computation and analyze the results in order to retrieve the secret key [4, 15, 17]. For fault detection of the encryption or decryption one may use redundant units [16]. Also, a multiplication approach for fault detection of the multiplicative inversion of the S-box and its inverse is presented in [15], where, the result of the multiplication of the input and the output of the multiplicative inversion is compared with the actual result. Another approach is the use of error detecting codes [2, 3, 5, 33, 34], for which, several parity prediction schemes are proposed. The output parity bits of each transformation in every round are used as inputs to the next transformation.

Parity prediction of the S-box used in [3] is based on the look-up table implementation in which memory cells are utilized to generate the predicted parity bit as well as the 8-bit output. The S-box and the inverse S-box as well as their parity predictions are nonlinear operations and are preferred to be implemented using look-up tables. The Look-up table-based S-boxes have 256 elements which are computed in advance and stored in the memory [5, 33]. However, for applications which require high performance AES implementations, one needs to implement the S-box and the inverse S-box in hardware using logic gates [7], especially when the high latency of the implementations is not an issue. The usage of arithmetic in the composite fields reduces the space complexity of the S-box and the inverse S-box and allows us to pipeline them which reduces the delay [14].

Instead of predicting the parity of the S-box, one may use an alternate implementation of the S-box and compare two outputs to obtain a fault detection scheme. Although using this method detects all errors at the output of the S-box, the area overhead, considering the alternate implementation and the comparison circuit, is more than 100% which is quite high for area-constrained applications. Another fault detection scheme may be based on comparing a number of output bits of the S-box, say n bits, with those of the alternate one using n two-input XOR gates. Then, using an n-input OR gate, the error indication flag is obtained. Using this scheme, only the faults that occur in the portion of circuit that generates the above mentioned n bits of the output in the original circuit may be detected. This means a non-uniform fault detection scheme which is dependent on the locations of the faults. However, this is not the case for the presented fault detection scheme based on the prediction of the parity bits of the output.

In this paper, we propose a general scheme for reaching fault detection structures of the AES S-boxes and inverse S-boxes. This scheme can be used for any hardware implementation of the S-boxes and the inverse S-boxes based on composite fields. Moreover, we implement fault detection AES structures using two commonly-used composite fields using polynomial basis [32, 35]. We utilize the logic gates implementation for the modified S-boxes and inverse S-boxes as well as their parity-based schemes. The contributions of this paper are summarized as follows:

- We have analyzed the error propagations due to all the single stuck-at faults in the structures of the S-boxes and the inverse S-boxes in two commonly used composite fields.
- After dividing the structures of the S-box and the inverse S-box into five blocks, we have proposed a general scheme to obtain modified structures of these blocks in any composite field realization of the S-boxes and the inverse S-boxes. Using these structures, single faults lead to either zero or odd number of erroneous output bits of the block in which the fault has occurred. Therefore, all the single faults are detected using the fault detection scheme.
- We have simulated the structures of the modified S-boxes and inverse S-boxes in two composite fields after injecting stuck-at faults. Our simulations using Quartus® II show that the fault detection schemes for the modified structures are capable of detecting all single faults. It is also shown that the fault detection schemes using modified structures can detect most of the random multiple faults.
- Finally, the ASIC synthesis results using the 0.18μ CMOS technology for the S-boxes, the inverse S-boxes, mixed S-boxes/inverse S-boxes and the AES encryption are presented. We have also synthesized the modified structures and then their areas, delays

and power consumptions are obtained. Furthermore, the complexities of the modified S-boxes and inverse S-boxes and their fault detection circuits are compared with the previous schemes.

The organization of this paper is as follows: In Section 2, preliminaries regarding the logic gate implementations of the S-boxes and the inverse S-boxes using composite fields $GF(((2^2)^2)^2)$ and $GF((2^4)^2)$ are explained. Also, their predicted parities are presented and the fault model used throughout this paper is introduced. In Section 3, error propagations due to single faults in the S-boxes and the inverse S-boxes in these composite fields are discussed. Fault detection structures of the modified S-boxes and inverse S-boxes for two above-mentioned composite fields are presented in Sections 4 and 5, respectively. Then, in Section 6, using the formulations of the parity predictions, parity-based fault detection schemes for those structures are simulated after injecting all single stuck and many random multiple faults. Moreover, the complexities of the proposed structures are analyzed in this section. In Section 7, the ASIC experimental results using the 0.18μ CMOS technology for the modified S-boxes and inverse S-boxes, mixed S-boxes/inverse S-boxes, and their fault detection schemes are presented. Also, we have implemented the structures of the entire AES encryption and their fault detection schemes using two composite fields. Their experimental results are also given in Section 7. Finally, conclusions are made in Section 8.

2 Preliminaries

The AES uses the irreducible polynomial of $p(z) = z^8 + z^4 + z^3 + z + 1$ to construct the binary field $GF(2^8)$. For obtaining low complexity implementations, composite fields are used [24]. In this section, we introduce two commonly used composite fields, i.e., $GF(((2^2)^2)^2)$ and $GF((2^4)^2)$, for implementation of the S-box and the inverse S-box. It is noted that although our presented scheme for obtaining the fault detection structures is applicable to any composite field, we have considered these fields since they have received much attention in the literature, see for example [26, 27, 29, 35] for $GF(((2^2)^2)^2)$ and [14, 18, 19, 25, 30, 32] for $GF((2^4)^2)$. Also, we introduce the parity predictions of the S-boxes and the inverse S-boxes using these composite fields and the fault model we use throughout this paper.

2.1 S-box and Inverse S-box in $GF(((2^2)^2)^2)$

The S-box structure which uses composite field arithmetics in $GF(((2^2)^2)^2)$ is shown in Fig. 1a [27]. As seen in this figure, for the composite field presented in [35], the transformation matrix $\boldsymbol{\delta}$ in Eq. 1, which is implemented by the first part of Block 1, transforms $X = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \in GF(2^8)$ to $\eta = \eta_h z + \eta_l \in GF(((2^2)^2)^2)$ using:

$$\boldsymbol{\eta} = \begin{pmatrix} \eta_0 \\ \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \\ \eta_5 \\ \eta_6 \\ \eta_7 \end{pmatrix} = \boldsymbol{\delta x} = \begin{pmatrix} 1\,1\,0\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1\,0\,1\,0 \\ 0\,1\,1\,1\,1\,0\,0\,1 \\ 0\,1\,1\,0\,0\,0\,1\,1 \\ 0\,1\,1\,1\,0\,1\,0\,1 \\ 0\,0\,1\,1\,0\,1\,0\,1 \\ 0\,1\,1\,1\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}, \quad (1)$$

where, $\eta_h = (\eta_7, \eta_6, \eta_5, \eta_4)$, $\eta_l = (\eta_3, \eta_2, \eta_1, \eta_0) \in GF((2^2)^2)$, and the polynomial $z^2 + z + \lambda$, $\lambda = (1100)_2$, is used to construct the composite field $GF(((2^2)^2)^2)$ over $GF((2^2)^2)$.

The output of Block 4 generates the multiplicative inversion of η, i.e., $\sigma = \sigma_h z + \sigma_l = \eta^{-1}$, where, $\sigma_h = (\sigma_7, \sigma_6, \sigma_5, \sigma_4) = \eta_h \theta$, $\sigma_l = (\sigma_3, \sigma_2, \sigma_1, \sigma_0) = (\eta_h + \eta_l)\theta$, and $\theta = (\eta_h^2 \lambda + \eta_h \eta_l + \eta_l^2)^{-1}$ are elements in the sub-field $GF((2^2)^2)$. It is noted that the fields $GF((2^2)^2)$ and $GF(2^2)$ are generated by $z^2 + z + \phi$ and $z^2 + z + 1$, respectively, where $\phi = (10)_2$. Block 5 in Fig. 1a implements the inverse transformation matrix, $\boldsymbol{\delta^{-1}}$, followed by an affine transformation which results in the output of the S-box, i.e., $Y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0) \in GF(2^8)$, as follows:

$$\boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,0\,0\,0\,0\,0\,0\,1 \\ 1\,0\,1\,1\,1\,1\,1\,0 \\ 1\,1\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,0\,0\,0\,1 \\ 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,0\,0\,0\,1 \end{pmatrix} \begin{pmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ \sigma_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}. \quad (2)$$

It is noted that all of the arithmetic operations including multiplications, inversion and squaring in Fig. 1a are in $GF((2^2)^2)$. Also, each addition is implemented by four two-input XOR gates (shown in Fig. 1a by shaded circles surrounding a plus).

As seen in Fig. 2a, the inverse S-box is implemented using the same multiplicative inversion as in the S-box, i.e., Blocks 2, 3 and 4 in Fig. 2a are the same as the corresponding blocks in Fig. 1a. However, Block 5

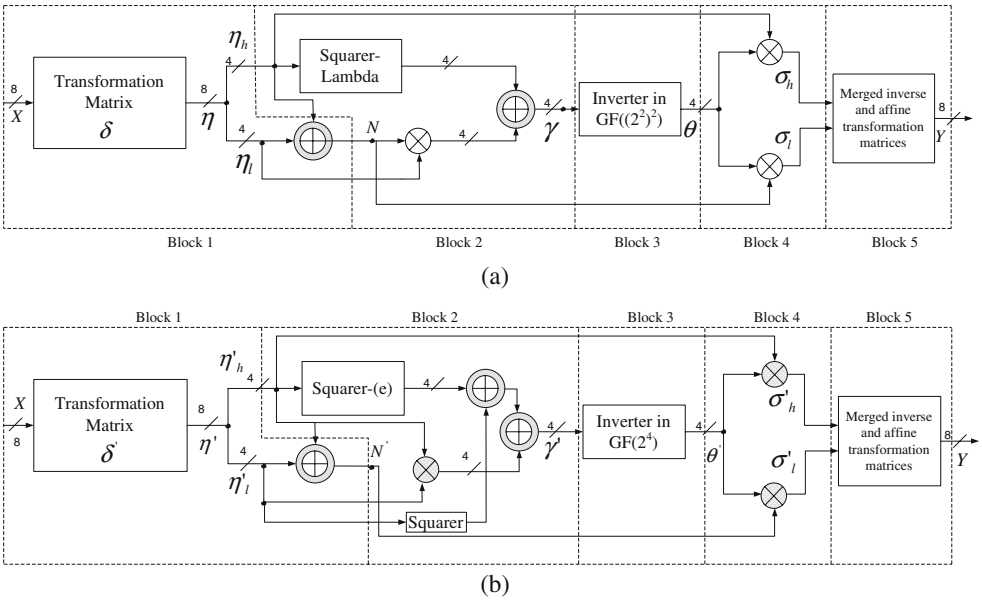

Fig. 1 Different blocks in the S-box using the composite field: **a** $GF(((2^2)^2)^2)$ [35] and **b** $GF((2^4)^2)$ [32]

consists of the inverse transformation matrix, $\boldsymbol{\delta^{-1}}$ as shown in Eq. 3. As seen in Fig. 2a, $\boldsymbol{\delta^{-1}}$ transforms $\eta = \eta_h z + \eta_l \in GF(((2^2)^2)^2)$ to $X = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \in GF(2^8)$ using:

$$\boldsymbol{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \boldsymbol{\delta^{-1}\eta} = \begin{pmatrix} 1\,0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,0\,0\,1\,1\,0\,0 \\ 0\,1\,1\,1\,1\,0\,0\,1 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,1\,1\,0\,1\,1\,1\,0 \\ 0\,1\,0\,0\,0\,1\,1\,0 \\ 0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,1\,1\,1 \end{pmatrix} \begin{pmatrix} \eta_0 \\ \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \\ \eta_5 \\ \eta_6 \\ \eta_7 \end{pmatrix}. \quad (3)$$

Moreover, as shown in Fig. 2a, the inverse affine transformation precedes the transformation matrix $\boldsymbol{\delta}$ and these merged transformations generate output σ of Block 1 from input Y as follows:

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ \sigma_7 \end{pmatrix} = \begin{pmatrix} 0\,0\,1\,0\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1\,0\,0 \\ 0\,1\,1\,0\,0\,1\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,1\,1\,1\,0\,0 \\ 1\,0\,0\,0\,1\,1\,1\,0 \\ 1\,1\,1\,1\,0\,0\,1\,1 \\ 0\,1\,1\,0\,0\,0\,1\,1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}. \quad (4)$$

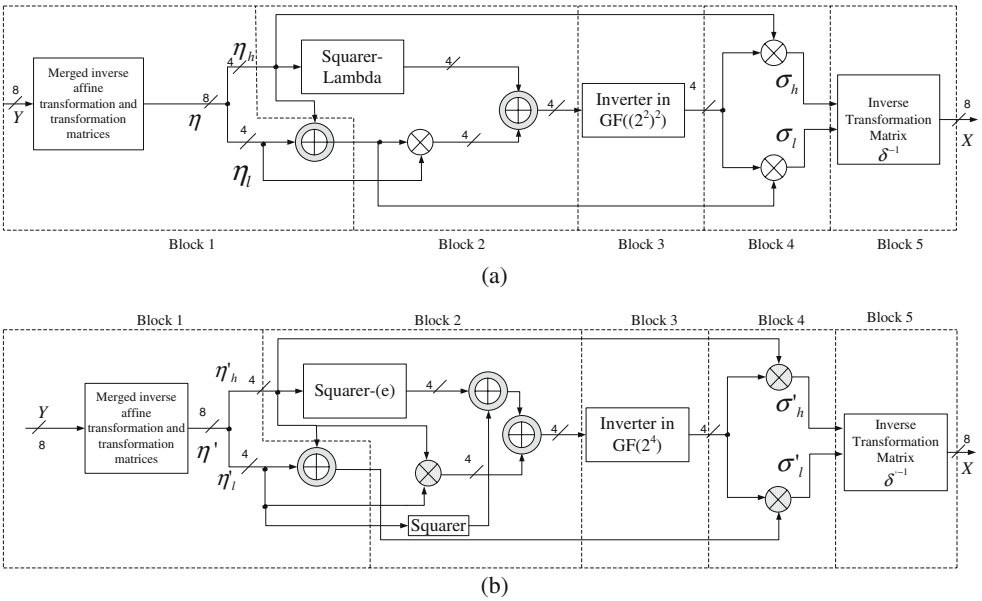

Fig. 2 Different blocks in the inverse S-box using the composite field: **a** $GF(((2^2)^2)^2)$ [35] and **b** $GF((2^4)^2)$ [32]

2.2 S-box and Inverse S-box Using $GF((2^4)^2)$

The implementation of the S-box in $GF((2^4)^2)$ is explained in [32] and is shown in Fig. 1b. Similar to the previous field, here, the transformation matrix $\boldsymbol{\delta'}$, which is presented in Eq. 5, transforms the input of the S-box, i.e., $X \in GF(2^8)$, to its representation $\eta' = \eta'_h z + \eta'_l \in GF((2^4)^2)$ as follows:

$$\boldsymbol{\eta'} = \begin{pmatrix} \eta'_0 \\ \eta'_1 \\ \eta'_2 \\ \eta'_3 \\ \eta'_4 \\ \eta'_5 \\ \eta'_6 \\ \eta'_7 \end{pmatrix} = \boldsymbol{\delta' x} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,1\,0 \\ 0\,1\,1\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0\,1 \\ 0\,0\,1\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1\,1\,1\,0 \\ 0\,1\,0\,0\,1\,0\,1\,1 \\ 0\,0\,1\,1\,0\,1\,0\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}, \quad (5)$$

where, similar to the previous composite field, $\eta'_h = (\eta'_7, \eta'_6, \eta'_5, \eta'_4)$ and $\eta'_l = (\eta'_3, \eta'_2, \eta'_1, \eta'_0)$ are sub-field representations of η' over $GF(2^4)$ which are represented with prime notations to distinguish them from the representations of elements in the composite field $GF(((2^2)^2)^2)$.

The irreducible polynomial of $z^2 + 1z + e$ with coefficients in $GF(2^4)$ is used for constructing $GF((2^4)^2)$ over $GF(2^4)$. Also, $z^4 + z + 1$ is the irreducible polynomial for operations over $GF(2^4)$ [32]. As shown in Fig. 1b, Block 5 implements the inverse transformation matrix $\boldsymbol{\delta'^{-1}}$ followed by an affine transformation. This results in the output of the S-box, i.e., Y, as:

$$\boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1\,0\,1\,0\,1\,1\,0\,1 \\ 1\,1\,1\,1\,1\,1\,0\,1 \\ 1\,0\,0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0\,1\,1 \\ 1\,1\,0\,1\,1\,0\,1\,1 \\ 0\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,1\,0\,1\,1 \\ 0\,1\,1\,0\,1\,0\,1\,1 \end{pmatrix} \begin{pmatrix} \sigma'_0 \\ \sigma'_1 \\ \sigma'_2 \\ \sigma'_3 \\ \sigma'_4 \\ \sigma'_5 \\ \sigma'_6 \\ \sigma'_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}. \quad (6)$$

As shown in Fig. 1b, the multiplicative inversion of this S-box consists of three multiplications in $GF(2^4)$ (shown by a cross surrounded by a shaded circle), an inversion in $GF(2^4)$, squaring and a constant multiplication. As seen in Fig. 2b, this multiplicative inversion, i.e., Blocks 2, 3 and 4, is also used for the inverse S-box. Similar to the previous field, here, $\boldsymbol{\delta'^{-1}}$ in the inverse S-box transforms $\eta' = \eta'_h z + \eta'_l \in GF((2^4)^2)$ to $X = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \in GF(2^8)$:

$$\boldsymbol{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \boldsymbol{\delta'^{-1}\eta'} = \begin{pmatrix} 1\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1\,0 \\ 0\,1\,0\,1\,1\,1\,0\,1 \\ 0\,0\,1\,0\,1\,1\,0\,0 \\ 0\,1\,1\,1\,1\,0\,0\,1 \\ 0\,0\,1\,0\,1\,1\,0\,1 \end{pmatrix} \begin{pmatrix} \eta'_0 \\ \eta'_1 \\ \eta'_2 \\ \eta'_3 \\ \eta'_4 \\ \eta'_5 \\ \eta'_6 \\ \eta'_7 \end{pmatrix}. \quad (7)$$

Also, from Fig. 2b, Block 1 in the inverse S-box consists of the inverse affine transformation preceding the transformation matrix $\boldsymbol{\delta'}$ in Eq. 5 to generate the output σ' from the input Y:

$$\boldsymbol{\sigma'} = \begin{pmatrix} \sigma'_0 \\ \sigma'_1 \\ \sigma'_2 \\ \sigma'_3 \\ \sigma'_4 \\ \sigma'_5 \\ \sigma'_6 \\ \sigma'_7 \end{pmatrix} = \begin{pmatrix} 0\,1\,1\,1\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1\,0\,1 \\ 1\,0\,1\,1\,0\,1\,1\,0 \\ 0\,1\,1\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,1\,1\,0\,0 \\ 0\,1\,1\,0\,0\,0\,1\,1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}. \quad (8)$$

2.3 Parity Predictions of S-boxes and Inverse S-boxes

In the fault detection schemes, the parity of each of 5 blocks in Fig. 1a, b (Fig. 2a, b) for the S-boxes (the inverse S-boxes) is predicted and it is compared with the actual parity. This method has been utilized in the literature to develop a fault detection scheme for different applications, see for example [3, 8, 9, 11, 23]. The comparison between the actual and predicted parities is implemented by XOR gates as shown in Fig. 3. As seen in this figure, depending on the number of input bits (either 4 or 8 in this paper), the actual parity is

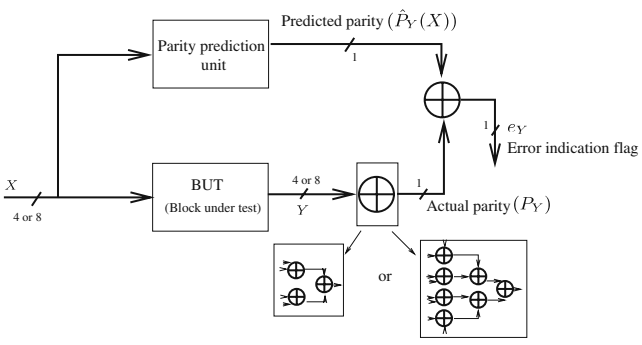

Fig. 3 Error indication for each block used in the fault detection scheme

obtained using the tree of XORs (shown in Fig. 3 by an XOR gate surrounded by a rectangle). Then, the predicted and actual parities of each block are compared to obtain the error indication flag. The predicted parities of the S-boxes and the inverse S-boxes for composite fields $GF(((2^2)^2)^2)$ and $GF((2^4)^2)$ presented before, are proved in [21]. These are presented below:

Theorem 1 [21] *The parity predictions of Blocks* 1 *to* 5 *of the S-box using* $GF(((2^2)^2)^2)$ *shown in Fig.* 1*a are obtained as follows:*

$$\hat{P}_N = \hat{P}_\eta = x_5 + x_4 + x_2 + x_0, \tag{9}$$

$$\hat{P}_\gamma = \eta_4 + \eta_3(\overline{P}_{\eta_h} + \eta_5) + \overline{\eta}_2(P_{\eta_h} + \eta_6) + \eta_1(\eta_6 + \eta_4) + \eta_0\overline{P}_{\eta_h}, \tag{10}$$

$$\hat{P}_\theta = (\overline{\gamma}_2 \vee \gamma_1)\gamma_0 + (\gamma_1 + \gamma_0)\gamma_3, \tag{11}$$

$$\hat{P}_\sigma = \eta_3(P_\theta + \theta_1) + \eta_2(P_\theta + \theta_2) + \eta_1(\theta_2 + \theta_0) + \eta_0 P_\theta, \tag{12}$$

$$\hat{P}_Y = \sigma_6 + \sigma_4 + \sigma_2 + \sigma_1 + \sigma_0. \tag{13}$$

where $\vee$ *represents the OR operation,* $P_{\eta_h} = \eta_7 + \eta_6 + \eta_5 + \eta_4$, *and* $P_\theta = (\theta_3 + \theta_2 + \theta_1 + \theta_0)$ *is the actual parity of Block* 3 *in Fig.* 1*a.*

For the inverse S-box, the predicted parities of Blocks 2, 3 *and* 4 *of the S-box in Fig.* 1*a are used to obtain the parity predictions of Blocks* 2, 3 *and* 4 *in Fig.* 2*a. Also, the parity predictions of Blocks* 1 *and* 5 *are obtained as follows:*

$$\hat{P}_\sigma = y_7 + y_6 + y_5 + y_3, \tag{14}$$

$$\hat{P}_X = \eta_6 + \eta_4 + \eta_2 + \eta_1 + \eta_0. \tag{15}$$

Theorem 2 [21] *The parity predictions of five blocks of the S-box using* $GF((2^4)^2)$ *shown in Fig.* 1*b are obtained as follows:*

$$\hat{P}_{N'} = \hat{P}_{\eta'} = x_6 + x_3 + x_2 + x_1 + x_0, \tag{16}$$

$$\hat{P}_{\gamma'} = \eta'_3\eta'_4 + \eta'_2(\eta'_5 + \eta'_4) + \eta'_1(\overline{P_{\eta'_h}} + \eta'_7) + \eta'_0\overline{P_{\eta'_h}} + P_{\eta'_h}, \tag{17}$$

$$\hat{P}_{\theta'} = \overline{\gamma'}_3\gamma'_2\overline{\gamma'}_0 + \gamma'_0(\overline{\gamma'_1} \vee \overline{(\gamma'_2 + \gamma'_3)}), \tag{18}$$

$$\hat{P}_{\sigma'} = \eta'_3\theta'_0 + \eta'_2(\theta'_1 + \theta'_0) + \eta'_1(P_{\theta'} + \theta'_3) + \eta'_0 P_{\theta'}, \tag{19}$$

$$\hat{P}_Y = \sigma'_7 + \sigma'_6 + \sigma'_2 + \sigma'_0. \tag{20}$$

where, $P_{\eta'_h} = \eta'_7 + \eta'_6 + \eta'_5 + \eta'_4$, *and* $P_{\theta'} = (\theta'_3 + \theta'_2 + \theta'_1 + \theta'_0)$ *is the actual parity of Block* 3 *in Fig.* 1*b.*

Similarly, $\hat{P}_{\gamma'}$, $\hat{P}_{\theta'}$ *and* $\hat{P}_{\sigma'}$ *in the S-box of Fig.* 1*b are used to obtain the parity predictions of Blocks* 2, 3 *and* 4 *in Fig.* 2*b. Moreover, the parity predictions of Blocks* 1 *and* 5 *are:*

$$\hat{P}_{\sigma'} = y_4 + y_2 + y_1 + y_0, \tag{21}$$

$$\hat{P}_X = \eta'_7 + \eta'_6 + \eta'_2 + \eta'_0. \tag{22}$$

It is noted that the correctness of the above formulations is also verified by simulations for all 256 input combinations [21].

2.4 Fault Model

In this paper, we use stuck-at faults model at the logic level. This type of fault forces one node (for single stuck-at fault model) or multiple nodes (for multiple stuck-at fault model) to be stuck at logic one (for stuck-at one) or zero (for stuck-at zero) independent of the fault-free logic values, see for example [20] and [28]. We have considered both single and multiple faults in this paper. For the single faults, only one node at a time becomes faulty, whereas, for the multiple faults, random multiple faults are injected where, the numbers, locations and types of the faults are randomly chosen. It is noted that in this paper we inject the faults at any node of the circuit, i.e., inputs, outputs and fan-out branches of the logic circuits. This type of fault injection has been used vastly, see for example [1].

In the fault attacks, the single faults injection is the ideal case through which the attackers gain more information. In this regard, we modify the structures of the S-box and its inverse so that all the single stuck-at faults are detected. However, because of not having precise attack tools due to technological constraints, a more realistic fault model is to inject multiple faults [6, 13]. Therefore, for covering both natural faults and fault attacks, in addition to single faults, multiple faults need to be considered [6]. Although the modified structures are presented to reach the fault coverage of close to 100%

for single faults, they are also capable of detecting most of the multiple faults.

The presented fault detection scheme in this paper is independent of the life time of the faults. Thus, both permanent and transient stuck-at faults lead to the same fault coverage utilizing the proposed scheme.

3 Error Propagation in S-boxes and Inverse S-boxes

The error propagation study is important for proposing fault detection schemes. In this section, we evaluate the error propagation in the S-boxes and the inverse S-boxes considering the single faults for the above-mentioned composite fields.

3.1 Error Propagation

As a result of a single fault, the output of the S-box (and the inverse S-box) may become erroneous. Because of the non-linear structure of the S-box (and the inverse S-box), the propagation of errors is random. To investigate the error propagation, we simulate the S-boxes and the inverse S-boxes presented in the previous section. We use single stuck-at fault model in which the input and output nodes of each logic gate, i.e., every node and branch in the circuit, is fixed by either zero or one and it is independent of the actual value of that node.

There are 159 gates in the original S-box and inverse S-box in $GF(((2^2)^2)^2)$, i.e., Figs. 1a and 2a, respectively. Also, the S-box and the inverse S-box in $GF((2^4)^2)$ shown in Figs. 1b and 2b have 180 and 174 gates, respectively. We have injected both single stuck-at 0 and 1 faults in the inputs and output of each gate for each and all combinations of the 8-bit inputs. Considering the use of two-input gates only, our search space is $(159 \times 3) \times 2 \times 256 - 8 \times 2 \times 256 = 240,128$ for the former and $(180 \times 3) \times 2 \times 256 - 8 \times 2 \times 256 = 272,384$ for the S-box and $(174 \times 3) \times 2 \times 256 - 8 \times 2 \times 256 = 263,168$ for the inverse S-box of the latter, where, the total number of nodes are shown in parentheses, 2 is for both stuck-at zero and stuck-at one faults, and 256 indicates all combinations of 8-bit inputs. Note that we exclude $8 \times 2 \times 256 = 4,096$ cases for the eight output nodes of the S-boxes and the inverse S-boxes. Our exhaustive search simulation results are shown in Table 1. As seen from the table, for the S-box and the inverse S-box in $GF_1 = GF(((2^2)^2)^2)$, out of 240,128 search space, $240,128 - 128,659 = 111,469$ and $240,128 - 130,777 = 109,351$ cases become erroneous, respectively. Also, for the S-box and the inverse S-box in $GF_2 = GF((2^4)^2)$, out of their corresponding search spaces, $272,384 - 179,422 = 92,962$ and $263,168 - 181,152 = 82,016$ cases become erroneous, respectively. The details of the error propagation for these cases in terms of the number (and percentage) of erroneous output bits are also shown in Table 1. This table shows the importance of having fault detection schemes for the implementation of the S-boxes and the inverse S-boxes using the composite fields. The table shows the fact that by injecting single faults to the gates of the S-box and the inverse S-box, the number of erroneous output bits is random because of the non-linear structures of the S-box and the inverse S-box. As seen from the column for zero-erroneous bits in the table, the S-box and the inverse S-box using composite field $GF_2 = GF((2^4)^2)$ have better responses due to injected single faults as compared with their counterparts using $GF_1 = GF(((2^2)^2)^2)$.

Some single faults may lead to even number of erroneous bits in the output of the blocks in the S-box/inverse S-box. Then, the parity-based fault detection scheme will not be able to detect such faults if one uses the predicted parity of the corresponding block.

Table 1 Number of cases and their percentages due to exhaustive search of single faults in terms of number of erroneous output bits of the S-box (SB) and the inverse S-box (ISB) in composite fields $GF_1 = GF(((2^2)^2)^2)$ and $GF_2 = GF((2^4)^2)$

Field	Search space	# of cases in terms of number of erroneous output bits								
		0	1	2	3	4	5	6	7	8
GF_1 (SB)	240,128 (100%)	128,659 (53.6%)	12,974 (5.4%)	22,568 (9.4%)	28,217 (11.8%)	17,437 (7.3%)	20,766 (8.6%)	7,848 (3.3%)	1,398 (0.6%)	261 (0.1%)
GF_1 (ISB)	240,128 (100%)	130,777 (54.5%)	12,165 (5.1%)	16,123 (6.7%)	22,160 (9.2%)	23,796 (9.8%)	23,457 (9.7%)	9,993 (4.2%)	1,407 (0.6%)	250 (0.1%)
GF_2 (SB)	272,384 (100%)	179,422 (65.9%)	5,756 (2.1%)	7,793 (2.9%)	15,568 (5.7%)	29,679 (10.8%)	26,455 (9.7%)	6,020 (2.2%)	1,211 (0.4%)	480 (0.2%)
GF_2 (ISB)	263,168 (100%)	181,152 (68.8%)	12,658 (4.7%)	10,065 (3.9%)	17,529 (6.7%)	18,229 (6.9%)	17,145 (6.5%)	4,374 (1.6%)	1,294 (0.5%)	725 (0.3%)

Detecting the single faults in the S-box and the inverse S-box using the parity-based fault detection scheme depends on how these structures are implemented. Logic gate implementations of these structures can be done so that all the single faults are detected. We denote fault detection structures of the S-box and the inverse S-box as the ones that make our proposed parity prediction scheme capable of detecting all single faults. We state the following lemma to derive a rule for designing fault detection structures for the S-boxes and the inverse S-boxes. It is noted that this lemma is general and can be applied to any composite fields. As a result, fault detection structures can be obtained for the composite field S-boxes and inverse S-boxes to detect all single stuck-at faults.

Lemma 1 *Consider a circuit which only consists of XOR (and XNOR) or NOT gates and the number of paths from each node in the circuit to the output bits is odd. Then, considering the stuck-at fault model we introduced in the previous section, the number of erroneous bits will be zero or odd for each single fault occurring in each node of the circuit.*

Proof Consider a block consisting of XOR (and XNOR) gates. Let $O \in \{0, 1\}$ be a node value in the circuit in the fault free situation. For an injected fault in node O, we have one of the following two cases:

(a) The fault does not change the original value of the node in which the fault is injected, i.e., O is error free. Then, the output of the block is correct and there is no erroneous bit.
(b) The injected fault changes the value of node O to $O \oplus 1$. Because inverting the input of an XOR (XNOR) gate inverts its output, we have erroneous outputs in each gate in the path that connects node O to the output of the block. This is because node O (whether it is in the stem or the branch) is the input of another gate and a change in O to $O \oplus 1$ causes the output of that gate to be inverted and thus the error propagates to the output of the block. If the number of the paths to the output of the block is odd then the output bits have odd number of errors. Therefore, the parity prediction scheme will be able to detect the fault. This discussion is true when we have NOT gates in the circuit, because NOT gates are XOR gates with one input always one.

□

In the next two sections, the fault detection structures for the S-box and the inverse S-box in composite fields $GF(((2^2)^2)^2)$ and $GF((2^4)^2)$ are explained. We use Lemma 1 for blocks 1 and 5 of the S-box and the inverse S-box. It is noted that a specific case of Lemma 1 is the case where no subexpression sharing is used. In other words, each output bit is implemented using logic gates which are not shared for deriving the other output bits. In such a case, which is used for blocks 2, 3 and 4, the number of paths from each node in the circuit to the output bits of the corresponding block is one.

4 Modified S-box and Inverse S-box Using $GF(((2^2)^2)^2)$

To comply with Lemma 1, we obtain fault detection structures of different blocks of the S-box and the inverse S-box in the two composite fields.

4.1 Transformation Matrix in Block 1 of the S-box

The transformation matrix δ can be implemented using 12 XOR gates [35]. To achieve a low complexity implementation, one needs to reuse subexpressions and one of the possible implementations is shown in Fig. 4a which implements the following formulations from Eq. 1:

$$\begin{aligned}
\eta_7 &= x_5 + x_7, & \eta_6 &= [x_2 + x_7] + [x_1 + x_6] + x_4 + x_3,\\
\eta_5 &= [x_3 + x_5] + [x_2 + x_7], & \eta_4 &= [x_3 + x_5] + [x_2 + x_7] + x_1,\\
\eta_3 &= [x_2 + x_7] + [x_1 + x_6], & \eta_2 &= [x_2 + x_7] + [x_1 + x_6] + x_4 + x_3 + x_6,\\
\eta_1 &= [x_1 + x_6] + x_4, & \eta_0 &= [x_1 + x_6] + x_0,
\end{aligned} \tag{23}$$

where, subexpressions, which are shown in brackets, are reused in Fig. 4a.

In terms of fault detection capability, a single stuck-at fault may change even number of output bits in Fig. 4a. Therefore, although the output is erroneous, the parity-based scheme cannot detect the fault. For example, if a fault occurs in the output of the XOR gate shown by × in Fig. 4a, two outputs, i.e., η_4 and η_5, are affected. This will mask the output parity and thus the fault is not detected. Based on Lemma 1, we modify Fig. 4a so that the number of paths from each node in the circuit to the output is odd. The proposed fault detection structure for transformation matrix δ is shown in Fig. 4b.

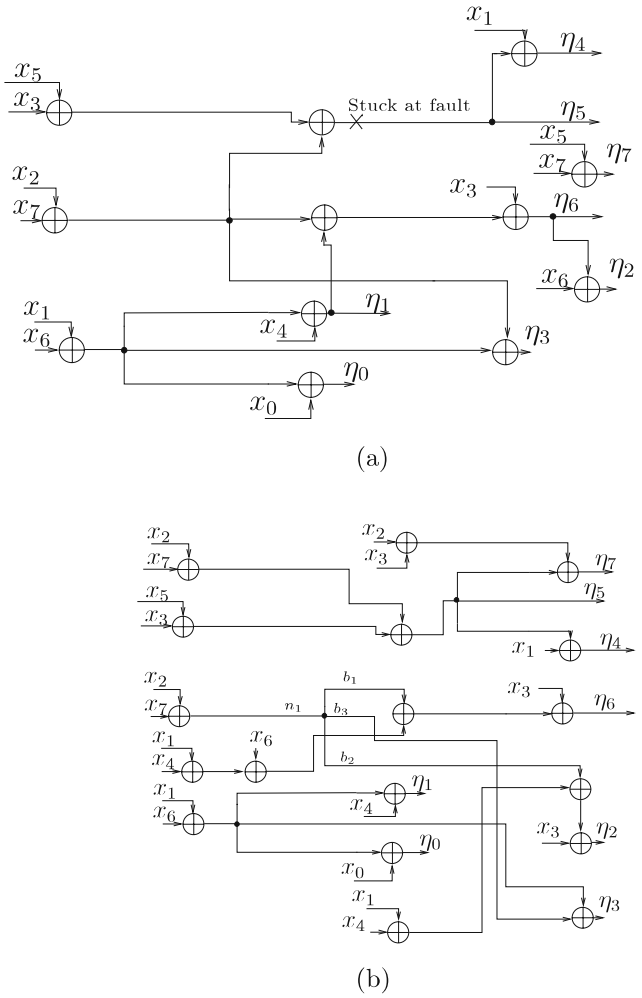

Fig. 4 Transformation matrix δ in Fig. 1a: **a** low complexity structure and **b** fault detection structure

The structure realizes the following formulations which are obtained from Eq. 23 as follows:

$$\begin{aligned}
\eta_7 &= ((x_2 + x_7) + (x_5 + x_3)) + (x_2 + x_3), & \eta_6 &= ((x_1 + x_4) + x_6 + (x_2 + x_7) + x_3),\\
\eta_5 &= ((x_2 + x_7) + (x_5 + x_3)), & \eta_4 &= ((x_2 + x_7) + (x_5 + x_3) + x_1),\\
\eta_3 &= ((x_2 + x_7) + (x_1 + x_6)), & \eta_2 &= ((x_1 + x_4) + (x_2 + x_7) + x_3),\\
\eta_1 &= ((x_1 + x_6) + x_4), & \eta_0 &= ((x_1 + x_6) + x_0).
\end{aligned} \tag{24}$$

Using the fault detection structure of Fig. 4b, any single fault in Block 1 leads to zero or odd number of erroneous bits in the output of block 1 in Fig. 1a and it will be detected by error indication flag of Block 1. It is noted that a single stuck-at fault at any node or branch in the modified structures can also be detected. As an example, a fault in the node n_1 in Fig. 4b causes three outputs, i.e., η_2, η_3 and η_6, to be affected and the error indication flag will detect it. Also, any single fault in branches b_1 to b_3 affects one output bit and this would be detected by the detection flag as well. As seen in Fig. 4b, the number of XOR gates is increased to 18. However, the propagation delay remains the same, i.e., $4T_X$, where T_X is the delay of an XOR gate.

4.2 Squarer-Lambda in the S-box and Inverse S-box

Similarly, some single faults injected in multiplication by λ and squarer in the circuit presented in [27] cannot be detected by the error indication flags. The squarer and λ units can be merged in order to modify them to be a fault detection structure. The merged unit shown in Fig. 1a is called *Squarer-Lambda* and has fewer gates than two separate units as presented in [27].

For the input of $\eta_h = (\eta_7, \eta_6, \eta_5, \eta_4)$ in $GF((2^2)^2)$, the result of this unit is

$$\eta_h{}^2\lambda = (\eta_6 + (\eta_5 + \eta_4), \eta_7 + \eta_4, \eta_7, \eta_7 + \eta_6) \tag{25}$$

and the corresponding circuit can be implemented from Eq. 25 using four XOR gates with $2T_X$ of the critical path delay. One can verify that this circuit satisfies Lemma 1 in which any single fault results in at most one error in the output of Eq. 25.

4.3 Multiplier in Blocks 2 and 4 of Figs. 1a and 2a

In order to have a compact design, one can utilize the formulations used for the multiplication in $GF((2^2)^2)$ presented in [27]. By adding extra gates (hardware redundancy) this multiplier is modified so that any single stuck-at fault is detected. Let $U = (u_3, u_2, u_1, u_0)$ and $V = (v_3, v_2, v_1, v_0)$ be the inputs of the multiplier. Then, the formulations for the fault detection multiplier are presented in Eq. 29 of Appendix 1 and the corresponding circuit is shown in Fig. 5. This circuit consists of AND gates in between arrays of XOR gates. One can see from this figure that the coordinates of output $Z = (z_3, z_2, z_1, z_0)$ are implemented so that single faults occurring in the circuit may only affect one coordinate. For example, consider a single fault occurring in the output of the XOR gate shown by × in Fig. 5. This fault may cause an error in the output bit z_3 depending on the values of inputs u_2 and u_3. If $u_2 = u_3 = 1$ or $u_2 = u_3 = 0$, the fault does not affect z_3 and the output is error free. Otherwise, z_3 will be an erroneous value and because only one bit is affected, the fault detection scheme detects it. It is noted that 21 XOR gates and 16 AND gates are needed in Fig. 5. Furthermore, the critical path delay is $4T_X + T_A$, where, T_X and T_A are delays for XOR and AND gates, respectively.

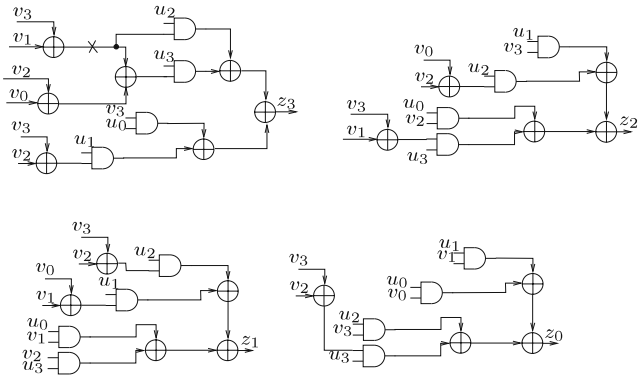

Fig. 5 Fault detection structure for multiplication in $GF((2^2)^2)$ in Blocks 2 and 4 of Figs. 1a and 2a

4.4 Block 3: Inversion in $GF((2^2)^2)$

We use the compact formulations proposed in [35] to obtain fault detection structure for the inversion in $GF((2^2)^2)$. Such formulations are presented in Eq. 30 of Appendix 1 and the corresponding circuit for inversion in $GF((2^2)^2)$ is shown in Fig. 6.

In this figure, an adder with a bar, i.e., $\bar{\oplus}$, is used to represent an XNOR gate. Since any fault in the nodes of Fig. 6 affects only one output, the number of erroneous output bits due to single faults is at most one. From Fig. 6, one can see that the inversion in $GF((2^2)^2)$ requires 6 XORs, 2 XNORs, 11 ANDs, 5 NOTs and 5 ORs. Moreover, the critical path delay is $2T_X + T_{XN} + T_A$, where T_{XN} is the delay of an XNOR gate.

4.5 Block 5: Merged Inverse and Affine Transformations of S-box

According to Lemma 1 and considering Eq. 2, the merged inverse and affine matrices, i.e., Block 5 in Fig. 1a, can be implemented so that this merged unit becomes a fault detection structure. The formulations for this fault detection structure using Eq. 2 are as follows:

$$
\begin{aligned}
y_7 &= (\sigma_{2,7} + \sigma_3), & y_6 &= \overline{((\sigma_4 + \sigma_5) + (\sigma_6 + \sigma_7))},\\
y_5 &= \overline{\sigma_{2,7}}, & y_4 &= (\sigma_{0,1} + (\sigma_4 + \sigma_7)),\\
y_3 &= (\sigma_{0,1} + \sigma_2), & y_2 &= (((\sigma_3 + \sigma_4) + (\sigma_5 + \sigma_6)) + (\sigma_0 + \sigma_2)),\\
y_1 &= \overline{(\sigma_0 + \sigma_7)}, & y_0 &= \overline{(\sigma_{2,7} + \sigma_{0,1} + \sigma_6)},
\end{aligned}
\tag{26}
$$

where, $\sigma_{2,7} = \sigma_2 + \sigma_7$ and $\sigma_{0,1} = \sigma_0 + \sigma_1$.

One can implement the circuit corresponding to Eq. 26 using 17 XOR gates and four NOT gates with the critical path of $3T_X + T_N$. It is noted that the output of the XOR gates $\sigma_{2,7}$ and $\sigma_{0,1}$ are used three times in Eq. 26. Therefore, a fault that causes an error at the output of each of these two gates generates three erroneous output bits, whereas, the fault in other gate outputs causes at most one error in the output of this block. As a result, all single faults can be detected by the parity-based fault detection scheme.

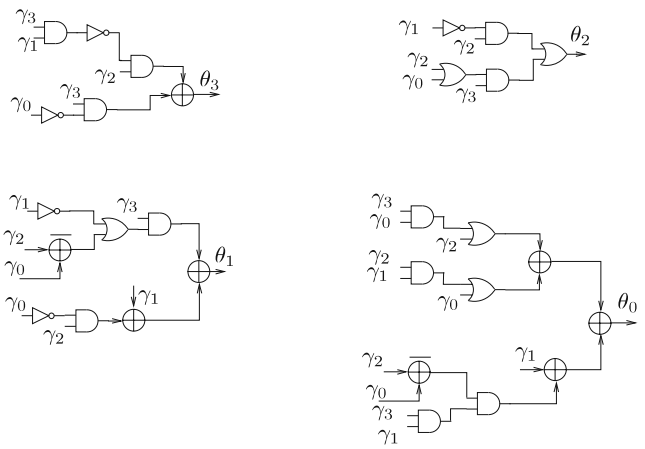

Fig. 6 Fault detection structure for Block 3 of Figs. 1a and 2a

4.6 Inverse S-box: Blocks 1 and 5

It is noted that Blocks 2, 3 and 4 in the inverse S-box are the same as Blocks 2, 3 and 4 in the S-box. Thus for the inverse S-box, we only need to obtain the fault detection structures for Blocks 1 and 5. According to Lemma 1 and considering Eqs. 3 and 4, Blocks 1 and 5 of the inverse S-box in Fig. 2a can be implemented so that these units become fault detection structures. The formulations for these two fault detection units are presented in Eqs. 31 and 32 of Appendix 1 and the circuits for blocks 1 and 5 of the inverse S-box are shown in Fig. 7a, b, respectively. As shown in Fig. 7a, the implementation of the merged inverse affine and transformation matrices (Block 1) needs 18 XOR gates and four NOT gates with the critical path delay of $3T_X$. Moreover, realizing the inverse transformation in $GF(((2^2)^2)^2)$, as shown in Fig. 7b, needs 19 XOR gates with $4T_X$ critical path delay.

5 Modified S-box and Inverse S-box Using $GF((2^4)^2)$

The fault detection structures of the S-box and its inverse in $GF((2^4)^2)$ are obtained in this section to comply with Lemma 1.

5.1 Transformation Matrix in Block 1 of the S-box

According to Lemma 1 and Eq. 5, we modify the low complexity transformation matrix presented in [32] so

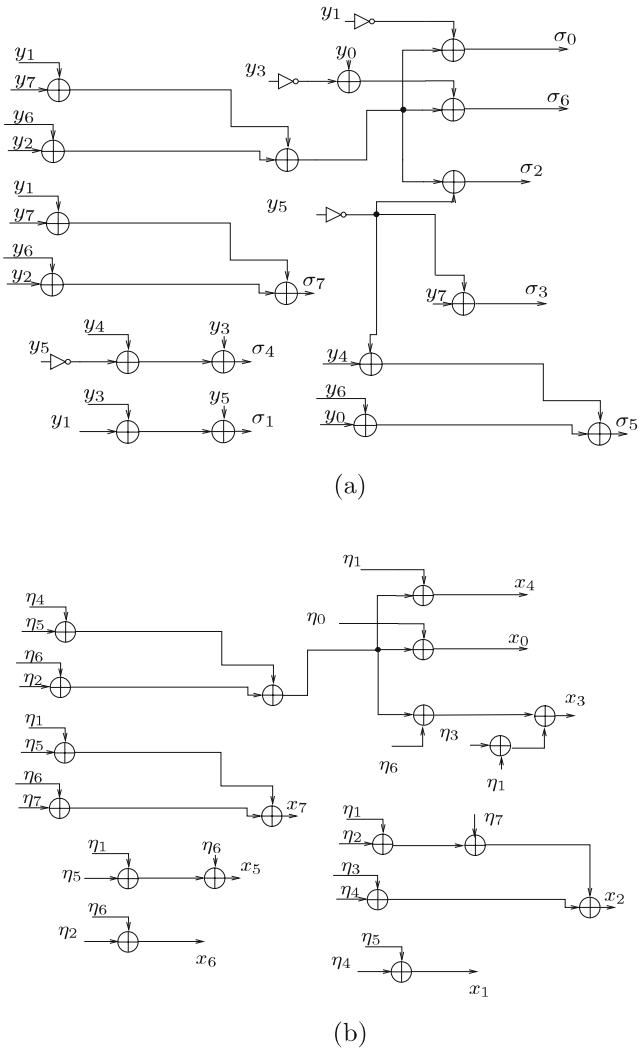

Fig. 7 Fault detection structures for two remaining blocks of the inverse S-box in Fig. 2a: **a** block 1: merged transformation and inverse affine transformation matrices and **b** block 5: inverse transformation

that the number of paths from each node to the output is odd and then propose a fault detection structure for the transformation matrix.

The formulations for the transformation matrix are presented in Eq. 33 of Appendix 2. The realization of Eq. 33 needs 13 XOR gates with the critical path delay of $3T_X$. This can be compared with 18 XOR gates with $4T_X$ in the critical path, needed for the transformation matrix in $GF(((2^2)^2)^2)$. Therefore, the realization of this structure needs less area and delay as compared with the one using $GF(((2^2)^2)^2)$.

5.2 Squarer-(e) in S-box and Inverse S-box

The squarer presented in [32] is a fault detection structure. Instead of applying squarer and constant multiplication by (e), we merge them and implement the merged unit so that it becomes a fault detection structure. The merged unit is called *Squarer*-(e) and has fewer gates than two separate units presented in [32]. For the input of $\eta'_h = (\eta'_7, \eta'_6, \eta'_5, \eta'_4)$ in $GF(2^4)$ the result of this unit is

$$\eta'^2_h(e) = (\eta'_6 + \eta'_5, \eta'_4, (\eta'_7 + \eta'_5) + \eta'_4, \eta'_5 + \eta'_4), \tag{27}$$

which needs four XOR gates for realization with $2T_X$ of critical path delay. These are the same as the area and the critical path delay of the *Squarer-Lambda* in the S-box and the inverse S-box in $GF(((2^2)^2)^2)$.

5.3 Multiplication and Inversion in S-box and Inverse S-box

The compact multiplication and inversion in $GF(2^4)$ presented in [32] are similarly modified so that they become fault detection structures. Let $U' = (u'_3, u'_2, u'_1, u'_0)$ and $V' = (v'_3, v'_2, v'_1, v'_0)$ be the inputs of the multiplier. Then, the coordinates of the output of the multiplier are obtained according to Eqs. 34 and 35 of Appendix 2, respectively. Using Eqs. 34 and 35, the corresponding circuits are shown in Fig. 8a, b, respectively. The circuit shown in Fig. 8a needs 18 XOR gates and 16 AND gates with the critical path delay of $3T_X + T_A$. This is three XOR gates less in area and one T_X less in delay in comparison with the multiplier in $GF((2^2)^2)$ presented in [35]. Furthermore, as seen in Fig. 8b, the inversion in $GF(2^4)$ needs 12 XORs, 5ORs, 9ANDs and one NOT gate with the critical path delay of $T_N + 3T_X + T_A$.

5.4 Merged Inverse and Affine Transformations (Block 5) of S-box

According to Eq. 6, the inverse and affine transformations in the S-box presented in [32] can be merged and modified so that the merged block becomes a fault detection structure. The formulations for the merged inverse and affine transformations (Block 5) of the S-box in Fig. 1b are not presented for the sake of simplicity. However, the circuit is shown in Fig. 9. As seen in the figure, 20 XOR gates and two NOT gates are needed for realizing the merged inverse and affine transformations block. Also, the critical path delay is $4T_X$. This needs three XOR gates more and two NOT gates less for implementations in comparison with the same block in $GF(((2^2)^2)^2)$.

It is interesting to note that overall, the area and delay of the fault detection S-box in $GF((2^4)^2)$ are less than the ones in $GF(((2^2)^2)^2)$. These are presented

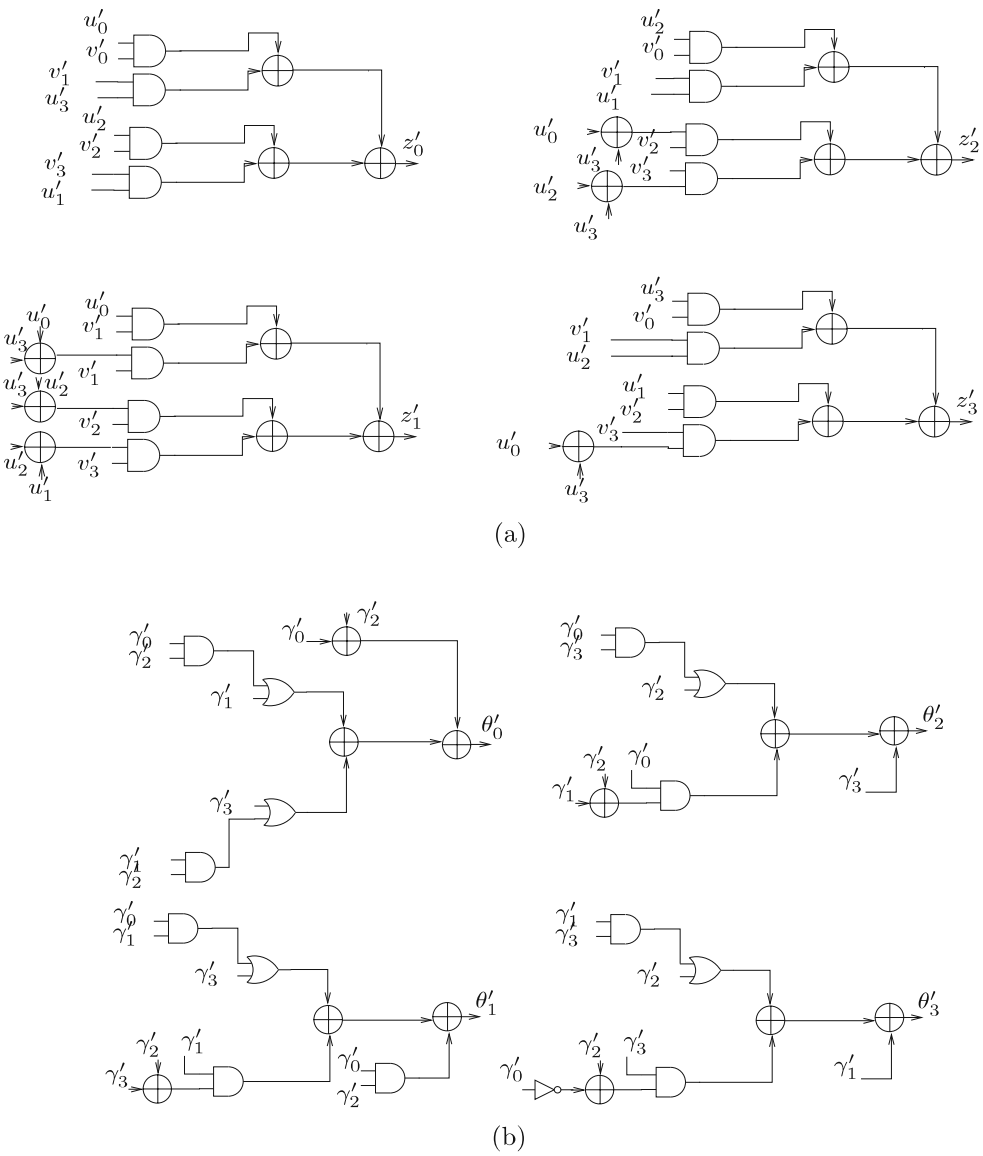

Fig. 8 Fault detection structures for:
a Multiplication in Blocks 2 and 4 of Figs. 1b and 2b and
b Block 3 of Figs. 1b and 2b

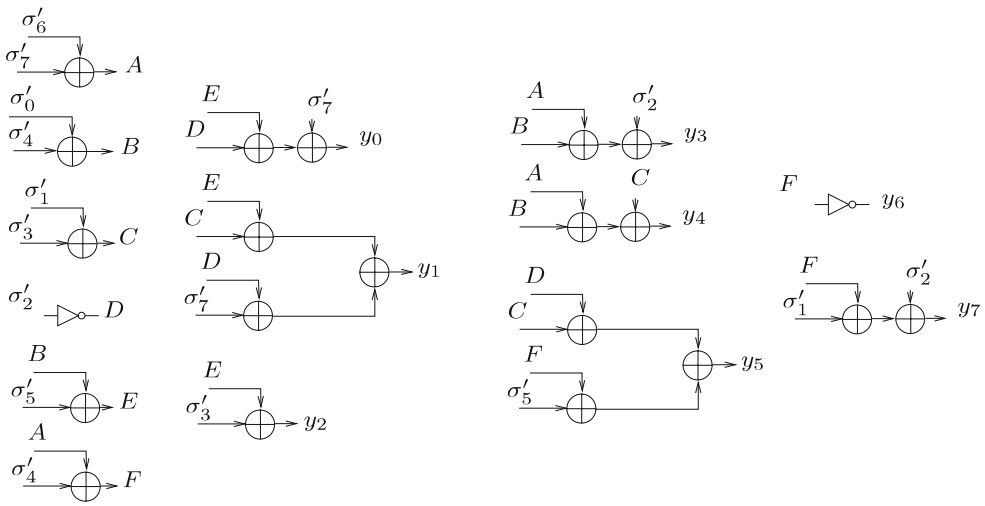

Fig. 9 Fault detection structure for Block 5 in Fig. 1b

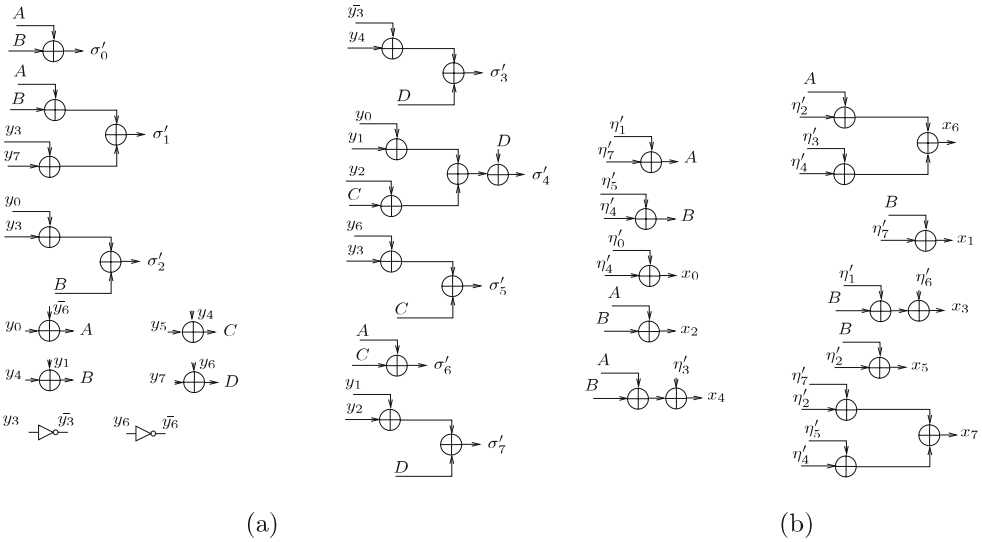

Fig. 10 Fault detection structures for two remaining blocks of the inverse S-box in Fig. 2b: **a** Block 1: Merged inverse affine and transformation matrices and **b** Block 5: Inverse transformation matrix

in details in Section 7 using the results of the ASIC experimental analysis.

5.5 Inverse S-box: Blocks 1 and 5

According to Eq. 7, the inverse affine transformation and the transformation matrix in the inverse S-box presented in [32] have been merged and modified so that the merged block becomes a fault detection structure. According to Eq. 8 and to comply with Lemma 1, the proposed implementation of inverse transformation is also modified to be a fault detection structure. The circuits for these two are shown in Fig. 10a, b, respectively. As seen in Fig. 10a, the inverse affine transformation and the transformation matrices need 21 XOR gates and two NOT gates with four XOR gates in the critical path. This needs two NOT gates less and three XOR gates more for implementations and one T_X more for delay in comparison with the same block in $GF(((2^2)^2)^2)$. For the inverse transformation in Fig. 10b, 16 XOR gates are required with three XOR gates in the critical path. This is three XOR gates less in area and one T_X less in the critical path delay comparing to the inverse transformation in $GF(((2^2)^2)^2)$.

Overall, the space complexity and the critical path delay of the fault detection inverse S-box in $GF((2^4)^2)$ is less than the one in $GF(((2^2)^2)^2)$. This is obtained in Section 7 utilizing the ASIC syntheses for these structures.

6 Parity-Based Fault Detection Schemes

In the fault detection schemes, the predicted parity of each block in the S-boxes (and the inverse S-boxes) is compared with its actual parity in order to obtain the error indication flag of the corresponding block. By ORing five indication flags of five blocks, the error indication of the entire S-boxes (and the inverse S-boxes) are obtained. We assume that this five-input OR gate is fault tolerant and one can easily implement it using logic gates resistant to natural and injected faults or utilizing triple modular redundancy [31]. This assumption increases the reliability of the fault detection implementations. However, if one does not consider this assumption and this OR gate becomes faulty, the error indication flag may generate an erroneous output bit. If the erroneous output bit of this OR gate is "0", i.e., in the fault free situation it generates "1", it does not indicate the occurrence of faults. On the other hand, if the erroneous output bit of this gate is "1", i.e., the fault free output of "0", this error indicator generates a false alarm. As a result, not considering the OR gate as a fault tolerant gate, decreases the fault coverage and increases the false alarms.

We have simulated the proposed modified S-boxes and inverse S-boxes for two composite fields by injecting single and multiple faults to all gates (except for the last five-input OR gate) for all combinations of 256 inputs and the simulation results are presented in this section.

6.1 Simulation Results Due to Single Faults

As discussed earlier, single faults occurring in the structures of the S-box (and the inverse S-box) which cause even number of erroneous output bits will not be detected if one compares the predicted parity of the corresponding block with its actual parity. Thus, in order to detect all single faults in each block using the

Table 2 Error propagation in the S-box (SB) and the inverse S-box (ISB) blocks in composite fields $GF_1 = GF(((2^2)^2)^2)$ and $GF_2 = GF((2^4)^2)$ due to single faults

Field	Structure	Number of erroneous bits	Block number				
			One	Two	Three	Four	Five
GF_1 (SB)	Original [35]	Even	3,072	10,854	3,046	12,072	6,540
		Odd	6,144	15,226	10,186	18,671	9,984
	Presented	Even	0	0	0	0	0
		Odd	9,216	28,608	10,959	49,872	13,824
GF_1 (ISB)	Original [35]	Even	4,536	10,854	3,046	12,072	5,230
		Odd	9,480	15,226	10,186	18,671	9,192
	Presented	Even	0	0	0	0	0
		Odd	16,896	28,608	10,959	49,872	14,592
GF_2 (SB)	Original [32]	Even	512	2,016	3,808	3,867	7,200
		Odd	1,982	25,776	16,284	30,570	18,432
	Presented	Even	0	0	0	0	0
		Odd	13,068	36,000	34,212	57,823	12,321
GF_2 (ISB)	Original [32]	Even	4,992	2,016	3,808	3,867	1,536
		Odd	13,056	25,776	16,284	30,570	9,180
	Presented	Even	0	0	0	0	0
		Odd	14,700	36,000	34,212	57,823	10,872

parity prediction scheme, we use modified blocks for the S-boxes and their inverses so that all single faults lead to zero or odd number of errors in the output of blocks. We have simulated the original S-boxes and inverse S-boxes and the modified ones proposed in this paper by injecting single faults for all combinations of 256 inputs.

The results of this simulation are depicted in Table 2. As seen in this table, the number of cases that result in even and odd number of erroneous output bits of each block is shown for both the original S-boxes/inverse S-boxes and the modified ones proposed here. As seen Table 2, the error indicator of the proposed fault detection scheme based on the modified S-boxes (inverse S-boxes) in two composite fields are able to detect all single faults. However, one can see from this table that this is not the case for the fault detection scheme based on the original S-boxes (inverse S-boxes) presented in [35] and [32].

6.2 Simulation Results Due to Multiple Faults

In the previous section, we have simulated the modified structures of the S-box and the inverse S-box and it is shown that all the single stuck-at faults can be detected. However, as stated before, due to the technological constraints, single stuck-at faults may not be applicable for an attacker. Therefore, in addition to single faults, multiple faults need to be considered in practice for covering both natural faults and fault attacks [6]. This is considered in this section. The presented schemes are capable of detecting a number of the multiple faults. It is noted that for our fault detection structures in the modified S-box and inverse S-box, multiple fault injection in the whole S-box and inverse S-box is considered. We have simulated our proposed S-boxes and inverse S-boxes implemented in both composite fields by injecting 256,000 random multiple faults in their structures. It is interesting to note that our simulations show that this number of injected faults is sufficient for reaching the actual results with a very good approximation. As a matter of fact, after injecting 2,560,000 faults (10 times as many as our selected number for fault injection) the obtained results are almost the same.

For random fault injection, linear feedback shift registers (LFSRs) within Quartus® II are utilized to distribute multiple faults randomly. In this regard, maximum sequence length polynomial for the feedback is selected according to the maximum sequence length taps (Altera: http://altera.com), [12]. Using the LFSR is an efficient method of generating pseudo random sequences (Altera: http://altera.com). It is also noted that as presented in this section, not only the simulation results comply with the theory but they are the same by increasing the number of injected faults from 256,000 to 2,560,000. This verifies the efficiency of the LFSRs used for random faults injection. We inject random faults in the sense that the locations, the number of gates that faults are injected in, and the types of faults either stuck-at zero or one are random. For each of the 256 possible inputs, one thousand random multiple faults (256,000 in total) are injected in the structures of five blocks of the modified S-boxes and inverse S-boxes and their parity prediction units as well as the actual parities. The distribution of the random faults in the blocks of the S-box and the inverse S-box is shown in Table 3. It is interesting to note that this distribution

Table 3 Multiple fault injection distribution in five blocks of the modified S-box (SB) and inverse S-box (ISB) in $GF_1 = GF(((2^2)^2)^2)$ and $GF_2 = GF((2^4)^2)$ due to injecting 256,000 random multiple faults

Composite field	Operation	Block 1 (%)	Block 2 (%)	Block 3 (%)	Block 4 (%)	Block 5 (%)
GF_1	SB	12	24	14	39	11
	ISB	13	23	15	38	11
GF_2	SB	10	24	15	39	12
	ISB	15	23	15	38	9

is dependent on the number of gates in these blocks. As seen in this table, for the S-box and inverse S-box, the distribution of the faults are shown. Moreover, for this distribution, the results of our simulations for such a distribution are shown in Table 4. In this table, N_1 is the number of cases whose errors are not detected by the error indication flag. Also, N_2 is the total number of cases where the final output is the same as the fault free output but the error indication flag detects an erroneous output. This is due to the faults in the parity prediction circuit or the faults that are masked in the output of the S-box/inverse S-box, i.e., false alarms. Moreover, N_3 is the number of detected errors, i.e., the faults that are detected using error indicator flags, and N_4 is the number of cases where faults do not result in any error. The fault coverage for the random multiple faults is obtained as fault coverage $= 100 \times \frac{N_3+N_2}{N_1+N_2+N_3}\%$ and is shown in Table 4.

The results of fault coverage for each S-box (inverse S-box) agree with the expression

$$100 \times \left(1 - \left(\frac{1}{2}\right)^5\right)\%. \tag{28}$$

It is noted that the probability of detecting (or not detecting) random multiple faults by the error indication flag of each block is $\frac{1}{2}$ and it is independent of those of other blocks. Thus, in Eq. 28, $(\frac{1}{2})^5$ is the probability that none of 5 blocks detects the injected faults, which results in the fault coverage as given in Eq. 28. In other words, five indicator flags are zeros while the output is erroneous.

As seen in Table 4, for each S-box (inverse S-box), 97% of the random faults are detected. According to [3], the detected errors at the output matrix state of SubBytes transformation consist of those that any of the fault detection circuits of the S-boxes can detect. In other words, for randomly distributed faults in the SubBytes, if at least one of the error indication flags of 16 S-boxes signals an error, the fault is detected. This is because the error indication flags of the S-boxes are ORed to obtain the error indication flag of the SubBytes.

Considering random faults occurring in 16 S-boxes (inverse S-boxes) in the SubBytes (inverse SubBytes) transformation, we present the following for the fault coverage of SubBytes (inverse SubBytes) transformation in the AES encryption (decryption).

Remark 1 Considering the fault coverage for one S-box/inverse S-box as 97% as presented in Table 4, the percentage of not detecting the randomly distributed faults in the SubBytes (inverse SubBytes) is $100 \times (\frac{3}{100})^{16}\% = (4.3 \times 10^{-23})\%$. This is negligible and can be considered zero with a very good approximation.

False Alarms

Our simulations show that the false alarms are a fraction of N_2. This fraction consists of the faults that do not cause erroneous output or erroneous parity bits but are falsely alarmed by the fault detection scheme. The false alarms is at most 0.14%, which can be neglected with a very good approximation for the total number of 256,000 injected faults. The details of the number of false alarms (out of 256,000) are obtained and tabulated in Table 4.

6.3 Complexity Analysis

The complexity analysis of the proposed modified S-boxes and inverse S-boxes using two composite fields $GF(((2^2)^2)^2)$ and $GF((2^2)^4)$ are presented in the following.

The overhead cost consists of the extra area and delay due to the parity-based fault detection schemes. Table 5 shows the area and the critical path delay of different sections of the modified S-boxes and inverse

Table 4 Fault Coverage (FC) and false alarms in the modified S-box (SB) and inverse S-box (ISB) in $GF_1 = GF(((2^2)^2)^2)$ and $GF_2 = GF((2^4)^2)$ due to injecting 256,000 random multiple faults in the whole structures

Composite field	Operation	N_1	N_2	N_3	N_4	False alarms	FC (%)
GF_1	SB	8,280	1,314	246,379	27	297 (0.12%)	≈ 97
	ISB	7,559	1,221	247,154	66	29 (0.13%)	≈ 97
GF_2	SB	8,612	1,532	245,832	24	347 (0.14%)	≈ 97
	ISB	8,241	1,293	246,387	79	281 (0.11%)	≈ 97

Table 5 Area and critical path delay of fault detection sections in the modified S-box (SB) and inverse S-box (ISB) in composite fields $GF_1 = GF(((2^2)^2)^2)$ and $GF_2 = GF((2^4)^2)$

$X = XOR$, $A = AND$, $XN = XNOR$, $N = NOT$, $O = OR$
T_X = Delay of an XOR, T_A = Delay of an AND, T_{XN} = Delay of an XNOR, T_N = Delay of a NOT, T_O = Delay of an OR

Composite field	Section	Area	Delay
GF_1	δ (SB)	$18X$	$4T_X$
	Squarer-Lambda	$4X$	$2T_X$
	Multiplication in $GF((2^2)^2)$	$21X + 16A$	$4T_X + T_A$
	Inversion in $GF((2^2)^2)$	$6X + 2XN + 5N + 5O + 11A$	$2T_X + T_{XN} + T_A$
	δ^{-1}+affine (SB)	$17X + 4N$	$3T_X + T_N$
	δ^{-1} (ISB)	$19X$	$4T_X$
	δ+inverse affine (ISB)	$18X + 4N$	$3T_X$
GF_2	δ' (SB)	$13X$	$3T_X$
	Squarer+(e)	$4X$	$2T_X$
	Multiplication in $GF(2^4)$	$18X + 16A$	$3T_X + T_A$
	Inversion in $GF(2^4)$	$12X + 5O + 9A + 1N$	$3T_X + T_A + T_N$
	δ'^{-1}+affine (SB)	$20X + 2N$	$4T_X$
	δ'^{-1} (ISB)	$16X$	$3T_X$
	δ'+inverse affine (ISB)	$21X + 2N$	$4T_X$

S-boxes for two composite fields. In this table, the area is presented in terms of the number of gates. The total overhead area cost of our parity-based fault detection schemes include the overhead of the modified S-boxes/inverse S-boxes, parity predictions, actual parity calculations and comparison circuits to generate error indication flags. For deriving the area overhead of the presented fault detection scheme, we assume that two-input AND and OR gates require 6 transistors each using the full CMOS technology. Also, two-input XOR and XNOR gates can be implemented using 10 transistors each [36] and a NOT gate can be realized using two transistors assuming that PMOS and NMOS need the same chip area. Table 6 shows the number of transistors needed for the original operations in [21], the modified structures based on the results in Table 5, and the actual and predicted parities and comparisons. As seen in this table, the area overhead for the fault detection S-box and inverse S-box over $GF(((2^2)^2)^2)$ are about 46% and 48%, respectively. This is because we need 1,582 and 1,602 transistors for implementing S-box and inverse S-box, respectively. Considering 1446 transistors for the original S-box and inverse S-box, 292 transistors for parity predictions and 240 transistors for the actual parities and comparison for both of them [21], one can reach the area overhead for this composite field. Also, the area overhead over $GF((2^4)^2)$ for the S-box and inverse S-box can be calculated similarly as 34% and 40%, respectively. As seen from the table, the modified structures over $GF((2^4)^2)$ have lower area overhead.

The delay overhead regarding the proposed schemes can overlap with the time required for the computations of other transformations after the S-boxes and inverse S-boxes. One can also take advantage of pipelining to minimize the delay overhead as follows. The parity predictions can take place in the current clock cycle and the actual parity calculation and comparison in the next clock cycle. Therefore, using pipelining, the delay of the fault detection scheme is one clock cycle, i.e., calculating the actual parity and error indication flag of Block 5 needs one extra clock cycle. In the pipelined AES, the encryption/decryption can continue to the next transformation while the parity prediction of the last block takes place.

7 ASIC Experimental Results

In this section, we explain the results of the syntheses we performed in two composite fields. We have used the 0.18μ CMOS technology for the syntheses of the S-boxes, inverse S-boxes, mixed S-boxes/inverse S-boxes and the whole fault detection structures of the AES.

Table 6 Area cost of the fault detection structures in terms of number of transistors in $GF_1 = GF(((2^2)^2)^2)$ and $GF_2 = GF((2^4)^2)$ for the S-box (SB) and inverse S-box (ISB)

Composite field	Operation	# for original	# for modified	# for parity predictions	# for actual parities and comparison	Total area overhead (%)
GF_1	SB	1,446	1,582	292	240	≈ 46
	ISB	1,446	1,602	292	240	≈ 48
GF_2	SB	1,544	1,548	280	240	≈ 34
	ISB	1,500	1,588	270	240	≈ 40

These structures have been coded in VHDL as the design entry to the Synopsys Design Analyzer™. We have selected the medium effort for the optimizations and obtained the timing, area and power reports. The syntheses details of the structures are explained in the following.

The original S-boxes and inverse S-boxes presented in [35] and [32] have been synthesized and their areas, delays and power consumptions are obtained. Then, these metrics are obtained considering the modified structures and the parity-based fault detection scheme. The results for two composite fields are shown in Table 7. As seen in the table, the area (μm^2), critical path delay (ns) and dynamic and leakage power consumptions (μW) are tabulated. It is noted that we have not used sub-pipelining for the above-mentioned structures and these syntheses are only intended to show the overheads of the presented schemes in this paper. In our syntheses the target frequency is 66 MHz (delay of 15 ns). As the table shows, for the fault detection architectures utilizing the modified S-boxes and inverse S-boxes, the delays are lower in most of the cases in comparison with the original structures. This is because in the modified structures, we have avoided using subexpression sharing which reduces the capacitor of the nodes that are reused. As a result, the actual delay of the fault detection structure is lower than the original ones.

We now compare the proposed fault detection schemes with the scheme for the original operations presented in [21], the redundant units in [16, 34], the multiplication approach in [15] and the parity-based scheme in [3] in terms of the fault coverage, the area and delay overheads. In [16] and [34], the S-box (the inverse S-box) is followed by its inverse and the original input is compared with the result of these consecutive operations to obtain the error indication flag. In the multiplication approach [15], the input and the output of the multiplicative inversion unit are multiplied and the result is compared to the predicted result. In [3], the parity-based fault detection scheme is based on the look-up table implementations in which 512×9 memory cells are used to generate the predicted parity bit as well as the 8-bit output. Table 8 shows the comparisons, where the results of Table 7 are used for the proposed fault detection schemes. As seen in Table 8, the single fault detection of the scheme for the original S-boxes and inverse S-boxes in [21] is around 80%. Also, it is noted that the simulations in [15] for $n = 2$ show that with the hardware overhead of 28.5%, the fault coverage is about 75%. Furthermore, for the redundant units approach the area and delay overhead are both around 100% which is quite high for high performance applications. Also, our ASIC experimental results for the scheme presented in [3] has been shown in this table. As seen in the table, the proposed fault detection structures are able to detect all the single faults and almost all multiple faults. It is interesting to note that, the working frequencies for the fault detection schemes using the modified S-boxes and inverse S-boxes are higher than those of the original operations, except for the S-box in GF_2, which is slightly lower than the original one (see the negative numbers for the critical path delay). As compared with the delays shown for the scheme for the original operations, the proposed fault detection structures have lower critical path delay.

Also, we have synthesized the original mixed S-boxes/inverse S-boxes presented in [35] and [32] which utilizes a common multiplicative inversion as well as the corresponding fault detection schemes. The structure uses multiplexers for selecting the operations and its corresponding fault detection circuits. The results for two composite fields are presented in Table 9. In the syntheses of the original mixed S-boxes/inverse S-boxes, we have considered two different structures presented in [35] and [32]. Then, we have used the modified S-boxes and inverse S-boxes as well as the fault detection circuit. Their results are shown in the last row of Table 9.

To obtain the overheads for the fault detection AES scheme, we have synthesized the original AES encryption [32, 35] and the fault detection AES encryption schemes using the modified S-boxes in the SubBytes transformation. For other encryption transformations, i.e., ShiftRows, MixColumns and AddRoundKey, the fault detection scheme presented in [3] is used. It is noted that we have not utilized sub-pipelining in these syntheses. However, registers are placed after each round of the AES encryption to increase the operating frequency. It is noted that the target frequency is 66 MHz (delay of 15 ns). The results of the syntheses for two composite fields are depicted in Table 10. As seen in the table, the area overhead of the fault detection scheme for two composite fields GF_1 and GF_2 are 34.3% and 32.6%, respectively. This overhead consists of the overheads for the modified structures for the S-box and the inverse S-box, fault detection circuits for these operations and for other transformations in the AES encryption. Furthermore, as seen in the table, the delay overhead for the AES encryption using modified S-box in GF_1 is negative which corresponds to higher maximum operating frequency of this structure.

Table 7 Comparing area, critical path delay, and power consumption of the proposed schemes for S-box (SB) and inverse S-box (ISB) in two composite fields

Structure	Area (μm^2)	Delay (ns)	Dynamic, leakage power consumption (μW)
Original SB GF_1 [35], GF_2 [32]	5415.4, 5423.5	13.5, 11.3	Dynamic: 493.2, 496.3 Leakage: 0.244, 0.244
Fault detection using modified SB GF_1, GF_2	7318.1, 7224.6	11.4, 11.4	Dynamic: 624.1, 606.9 Leakage: 0.352, 0.352
Original ISB GF_1 [35], GF_2 [32]	5382.9, 5321.9	12.4, 12.0	Dynamic: 482.3, 471.2 Leakage: 0.246, 0.236
Fault detection using modified ISB GF_1, GF_2	7212.4, 6907.1	12.1, 10.5	Dynamic: 619.3, 564.3 Leakage: 0.343, 0.340

Table 8 Comparing area, critical path delay, and fault coverage of the fault detection schemes using the modified S-box (SB) and inverse S-box (ISB) with other fault detection schemes

	Area overhead	Critical path delay overhead	Fault coverage
Proposed fault detection schemes GF_1, GF_2	35.1%, 33.2% (SB) 34.1%, 29.8% (ISB)	−15.5%, 0.8% (SB) −2.4%, −12.5% (ISB)	100% single[a] ≈ 97% multiple (for one SB/ISB)
Scheme for the original operations [21] GF_1, GF_2	26.3%, 23.5%(SB) 26.4%, 23.4%(ISB)	3.9%, 4.1% (SB) 3.5%, 3.9% (ISB)	≈ 73%, 87% (SB) single ≈ 74%, 88% (ISB) single ≈ 97% multiple
Redundant units [16], United S-box [34]	≈ 100%	≈ 100%	100%
Multiplication approach [15]	≈ 28.5% (SB)	Not mentioned	≈ 75%
Parity-based scheme in [3]	≈ 123% (SB)	≈ 7.3%	≈ 50%(random faults in one SB) (512 × 9 LUT)

[a]Assuming that the five-input OR gate used for deriving the final error indication flag is fault tolerant

Table 9 Comparing area, critical path delay, and power consumption of the proposed schemes using the mixed S-box (SB) and inverse S-box (ISB) in two composite fields

Structure	Area (μm^2)	Delay (ns)	Area and delay overhead	Power consumption (μW)	Dynamic, leakage power overhead
Mixed structures using SB & ISB GF_1 [35], GF_2 [32]	6,879.3, 7,033.4	14.1, 13.5	Area: 0%, 0% Delay: 0%, 0%	Dynamic: 701.2, 719.8 Leakage: 0.329, 0.319	0%, 0% 0%, 0%
Fault detection mixed structure using modified SB & ISB GF_1, GF_2	8,952.1, 8,928.0	12.0, 14.0	Area: 30.1%, 27.0% Delay: −14.8%, 3.3%	Dynamic: 889.2, 876.4 Leakage: 0.432, 0.421	26.8%, 21.9% 31.3%, 32.0%

Table 10 Comparing area, delay, and power consumption of the proposed schemes for the AES encryption

Structure	Area (μm^2)	Delay (ns)	Area and delay overhead	Power consumption (μW)	Dynamic, leakage power overhead
AES encryption using SB GF_1 [35],GF_2 [32]	974,351, 976,800	14.8, 13.5	Area: 0%, 0% Delay: 0%, 0%	Dynamic: 85,952, 87,612 Leakage: 44.7, 44.7	0%, 0% 0%, 0%
Fault detection AES encryption using modified SB GF_1, GF_2	1,308,172, 1,295,243	13.3, 13.6	Area: 34.3%, 32.6% Delay: −10.1%, 0.7%	Dynamic: 112,553, 111,014 Leakage: 64.5, 66.1	30.9%, 26.7% 44.3%, 47.9%

8 Conclusions

In this paper, we have considered parity-based fault detection schemes of the composite field realizations of the S-box and the inverse S-box for the advanced encryption standard. Our simulations show that using the parity-based fault detection schemes and the proposed modified S-box and inverse S-box structures, all single faults have been detected. We have also injected a large number of random multiple faults and our simulations show that the proposed scheme is able to detect almost all of the them with a low number of false alarms.

We have also synthesized the S-boxes, the inverse S-boxes and mixed S-boxes/inverse S-boxes for both original and modified structures as well as the AES encryption for the original and fault detection structures using the modified S-boxes and inverse S-boxes. The results of our ASIC syntheses show that the total area cost of the proposed fault detection schemes is less than that of the redundant units [16] and the united S-box [34] which have almost the same fault coverage as the presented schemes. Also, the fault coverage of our fault detection scheme is higher than the schemes presented in [3, 15] and the one in [21] for the original operations. It is interesting to note that in most of the cases, the working frequencies of the presented fault detection structures are higher than those for the original operations.

Acknowledgments A preliminary version of parts of this article was presented at IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'06) [22]. This work has been supported in part by an NSERC Discovery grant awarded to A. Reyhani-Masoleh.

Appendix 1

Formulations for Modified S-box and Inverse S-box in $GF(((2^2)^2)^2)$

Multiplier in Blocks 2 and 4 of Figs. 1a and 2a

Adding extra gates (hardware redundancy) the fault detection multiplier in $GF((2^2)^2)$, presented in Section 4, can be realized as follows:

$$\begin{aligned} z_3 &= (u_3((v_3+v_1)+(v_2+v_0))+u_2(v_3+v_1)) \\ &\quad +(u_1(v_3+v_2)+u_0v_3), \\ z_2 &= (u_3(v_3+v_1)+u_0v_2)+(u_2(v_2+v_0)+u_1v_3), \\ z_1 &= (u_3v_2+u_0v_1)+u_2(v_3+v_2)+u_1(v_1+v_0), \\ z_0 &= (u_3(v_3+v_2)+u_2v_3)+(u_1v_1+u_0v_0), \end{aligned} \tag{29}$$

where, $U=(u_3,u_2,u_1,u_0)$ and $V=(v_3,v_2,v_1,v_0)$ are the inputs and $Z=(z_3,z_2,z_1,z_0)$ is the output of the multiplier.

Block 3: Inversion in $GF((2^2)^2)$

The modified inversion in $GF((2^2)^2)$, presented in Section 4, can be implemented using the following formulations:

$$\begin{aligned} \theta_3 &= \gamma_2(\overline{\gamma_1\gamma_3})+\overline{\gamma}_0\gamma_3, \\ \theta_2 &= \overline{\gamma}_1\gamma_2 \vee \gamma_3(\gamma_0 \vee \gamma_2), \\ \theta_1 &= (\overline{\gamma}_0\gamma_2+\gamma_1)+\gamma_3(\overline{\gamma}_1 \vee \gamma_0\overline{+}\gamma_2), \\ \theta_0 &= ((\gamma_0\gamma_3 \vee \gamma_2)+(\gamma_1\gamma_2 \vee \gamma_0))+(\gamma_1+(\gamma_0\overline{+}\gamma_2)(\gamma_1\gamma_3)), \end{aligned} \tag{30}$$

where, $\overline{+}$ denotes an XNOR gate.

Inverse S-box

The merged inverse affine and transformation and the inverse transformation matrix are realized using the following formulations, respectively:

$$\begin{aligned} \sigma_7 &= (y_1+y_7)+(y_2+y_6), & \sigma_6 &= ((y_1+y_7)+(y_2+y_6)+(\overline{y_3}+y_0)), \\ \sigma_5 &= (y_4+\overline{y_5})+(y_0+y_6), & \sigma_4 &= (y_3+(y_4+\overline{y_5})), \\ \sigma_3 &= (y_7+\overline{y_5}), & \sigma_2 &= (y_1+y_7)+(y_2+y_6)+\overline{y_5}, \\ \sigma_1 &= ((y_1+y_3)+y_5), & \sigma_0 &= ((y_1+y_7)+(y_2+y_6)+\overline{y_1}). \end{aligned} \tag{31}$$

$$\begin{aligned} x_7 &= (\eta_1+\eta_5)+(\eta_7+\eta_6), & x_6 &= (\eta_2+\eta_6), \\ x_5 &= (\eta_1+\eta_5)+\eta_6, & x_4 &= (\eta_4+\eta_5)+(\eta_2+\eta_6)+\eta_1, \\ x_3 &= (\eta_4+\eta_5)+(\eta_2+\eta_6)+\eta_6+(\eta_1+\eta_3), & x_2 &= (\eta_1+\eta_2)+(\eta_3+\eta_4)+\eta_7, \\ x_1 &= (\eta_4+\eta_5), & x_0 &= (\eta_4+\eta_5)+(\eta_2+\eta_6)+\eta_0. \end{aligned} \tag{32}$$

It is noted that blocks 2, 3 and 4 of the inverse S-box are realized using those of the S-box presented in Section 4.

Appendix 2

Formulations for Modified S-box and Inverse S-box in $GF((2^4)^2)$

Transformation Matrix in Block 1 of the S-box

The formulations for the transformation matrix in $GF((2^4)^2)$ corresponding to the one presented in Section 5 is as follows:

$$\begin{aligned}
\eta'_7 &= [x_5 + x_7], & \eta'_6 &= (x_2 + x_3) + \{x_5 + x_7\},\\
\eta'_5 &= x_{4,6} + \{x_1 + x_7\}, & \eta'_4 &= (x_{4,6} + x_5),\\
\eta'_3 &= (x_2 + x_4), & \eta'_2 &= [x_1 + x_7],\\
\eta'_1 &= (x_1 + x_2), & \eta'_0 &= x_{4,6} + (x_0 + x_5),
\end{aligned} \tag{33}$$

where, $x_{4,6} = x_4 + x_6$ which has been used 3 times. Therefore, a fault at the output of this gate may result in zero or three erroneous output bits which will be detected using the fault detection scheme. Moreover, similar expressions denoted by {} and [ ] are not reused and are implemented using two different XOR gates.

Multiplication and Inversion in S-box and Inverse S-box

For the 4-bit inputs of U' and V', the coordinates of the 4-bit output Z' of the modified multiplier in $GF((2^4)$ introduced in Section 5 are obtained as follows:

$$\begin{aligned}
z'_3 &= (u'_2v'_1 + u'_3v'_0) + (v'_2u'_1 + v'_3(u'_0 + u'_3)),\\
z'_2 &= (u'_2v'_0 + u'_1v'_1) + (v'_2(u'_0 + u'_3) + v'_3(u'_2 + u'_3)),\\
z'_1 &= (u'_0v'_1 + v'_1(u'_3 + u'_0)) + (v'_2(u'_2 + u'_3) + v'_3(u'_1 + u'_2)),\\
z'_0 &= (u'_0v'_0 + u'_3v'_1) + (u'_2v'_2 + u'_1v'_3).
\end{aligned} \tag{34}$$

Let $Y' = (y'_3, y'_2, y'_1, y'_0)$ be the input of the inverter in $GF(2^4)$. Then, the coordinates of the output of inverter are

$$\begin{aligned}
\theta'_3 &= (\gamma'_1\gamma'_3 \vee \gamma'_2) + \gamma'_3(\overline{\gamma'_0} + \gamma'_2) + \gamma'_1,\\
\theta'_2 &= (\gamma'_0\gamma'_3 \vee \gamma'_2) + \gamma'_0(\gamma'_1 + \gamma'_2) + \gamma'_3,\\
\theta'_1 &= (\gamma'_0\gamma'_1 \vee \gamma'_3) + \gamma'_1(\gamma'_2 + \gamma'_3) + \gamma'_0\gamma'_2,\\
\theta'_0 &= (\gamma'_0\gamma'_2 \vee \gamma'_1) + (\gamma'_1\gamma'_2 \vee \gamma'_3) + (\gamma'_0 + \gamma'_2).
\end{aligned} \tag{35}$$

References

1. Abramovici M, Breuer MA, Friedman AD (1990) Digital systems testing and testable design. IEEE, Piscataway
2. Bertoni G, Breveglieri L, Koren I, Maistri P (2004) An efficient hardware-based fault diagnosis scheme for AES: performances and cost. In: Proc. of DFT 2004. IEEE Computer Society, Los Alamitos, pp 130–138
3. Bertoni G, Breveglieri L, Koren I, Maistri P, Piuri V (2003) Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. IEEE Trans Comput 52(4):492–505 (special issue on Cryptographic Hardware and Embedded Systems)
4. Boneh D, DeMillo RA, Lipton RJ (2001) On the importance of eliminating errors in cryptographic computations. J Cryptol 14(2):101–119
5. Breveglieri L, Koren I, Maistri P (2005) Incorporating error detection and online reconfiguration into a regular architecture for the advanced encryption standard. In: Proc. of DFT 2005. IEEE Computer Society, Los Alamitos, pp 72–80
6. Breveglieri L, Koren I, Maistri P (2007) An operation-centered approach to fault detection in symmetric cryptography ciphers computers. IEEE Trans Comput 56(5): 635–649
7. Canright D (2005) A very compact S-Box for AES. In: Rao JR, Sunar B (eds) CHES 2005. LNCS, vol 3659. Springer, Heidelberg, pp 441–455
8. Cardarilli GC, Ottavi M, Pontarelli S, Re M, Salsano A (2006) Fault localization, error correction, and graceful degradation in radix 2 signed digit-based adders. IEEE Trans Comput 55(5):534–540
9. Cardarilli GC, Pontarelli S, Re M, Salsano A (2005) A self checking reed Solomon encoder: design and analysis. In: Proc. of DFT 2005. IEEE Computer Society, Los Alamitos, pp 111–119
10. Federal Information Processing Standards Publication "197" (2001) Advanced encryption standard (AES). NIST, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
11. Fenn S, Gossel M, Benaissa M, Taylor D (1998) On-line error detection for bit-serial multipliers in $GF(2^m)$. J Electron Testing: Theory Appl 13(1):1998
12. George M, Alfke P (2007) Linear feedback shift registers in virtex devices. Xilinx application note 210. Available at http://www.xilinx.com/support/documentation/application_notes/xapp210.pdf
13. Giraud C (2004) DFA on AES. In: 4th international conference of AES (AES 2004), pp 27–41
14. Hodjat A, Verbauwhede I (2006) Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors. IEEE Trans Comput 55(4):366–372
15. Karpovsky M, Kulikowski KJ, Taubin A (2004) Differential fault analysis attack resistant architectures for the advanced encryption standard. In: CARDIS 2004, 153. Kluwer, Toulouse, pp 177–192
16. Karri R, Wu K, Mishra P, Yongkook K (2001) Fault-based side-channel cryptanalysis tolerant Rijndael symmetric block cipher architecture. In: Proc. of DFT 2001, pp 418–426
17. Malkin TG, Standaert F-X, Yung M (2006) A comparative cost/security analysis of fault attack countermeasures. In: Proc. of FDTC 2006, pp 159–172
18. Mangard S, Aigner M, Dominikus S (2003) A highly regular and scalable AES hardware architecture. IEEE Trans Comput 52(4):483–491
19. Mangard S, Pramstaller N, Oswald E (2005) Successfully attacking masked AES hardware implementations. In: Rao JR, Sunar B (eds) CHES 2005. LNCS, vol 3659. Springer, Heidelberg, pp 157–171
20. Monnet Y, Renaudin M, Leveugle R (2006) Designing resistant circuits against malicious faults injection using asynchronous logic. IEEE Trans Comput 55(9):1104–1115

21. Mozaffari-Kermani M (2007) Fault detection schemes for high performance VLSI implementations of the advanced encryption standard. M.E.Sc. thesis, Department of Electrical and Computer Engineering, The University of Western Ontario, London, Ontario, Canada. Available at: http://publish.uwo.ca/~mmozaff/Thesis.pdf
22. Mozaffari-Kermani M, Reyhani-Masoleh A (2006) Parity-based fault detection architecture of S-box for advanced encryption standard. In: Proc. of DFT 2006. IEEE Computer Society, Los Alamitos, pp 572–580
23. Reyhani-Masoleh A, Hasan MA (2006) Fault detection architectures for field multiplication using polynomial bases. IEEE Trans Comput 55(9):1089–1103
24. Rijmen V (2000) Efficient implemenation of the Rijndael S-box. Available at: http://www.iaik.tugraz.at/RESEARCH/krypto/AES/old/~rijmen/rijndael/sbox.pdf
25. Rudra A, Dubey PK, Jutla CS, Kumar V, Rao JR, Rohatgi P (2001) Efficient Rijndael encryption implementation with composite field arithmetic. In: Proc. of CHES 2001, pp 171–184
26. Satoh A, Morioka S (2002) An optimized S-Box circuit architecture for low power AES design. In: Proc. of CHES 2002, LNCS, vol 2523, pp 172–186
27. Satoh A, Morioka S, Takano K, Munetoh S (2001) A compact Rijndael hardware architecture with S-Box optimization. In: Boyd C (ed) ASIACRYPT 2001. LNCS, vol 2248, pp 239–254
28. Skorobogatov SP, Anderson RJ (2002) Optical fault induction attacks. In: Proc. of CHES 2002, pp 2–12
29. Standaert FX, Rouvroy G, Quisquater JJ, Legat JD (2003) Efficient implementation of Rijndael encryption in reconfigurable hardware: improvements and design tradeoffs. In: Walter CD, Koç ÇK, Paar C (eds) CHES 2003. LNCS, vol 2779. Springer, Heidelberg, pp 334–350
30. Trichina E (2003) Combinational logic design for AES subbyte transformation on masked data. In: Cryptology eprint archive: Report 2003/236, IACR, (http://eprint.iacr.org/), Report 2003/236
31. Von Neumann J (1956) Probabilistic logics and synthesis of reliable organizms from unreliable components. In: Automata studies. Princeton Univ. Press, Princeton, pp 43–98
32. Wolkerstorfer J, Oswald E, Lamberger M (2002) An ASIC implementation of the AES SBoxes. In: Preneel B (ed) CT-RSA 2002. LNCS, vol 2271. Springer, Heidelberg, pp 67–78
33. Wu K, Karri R, Kuznetsov G, Goessel M (2004) Low cost concurrent error detection for the advanced encryption standard. In: Proc. of international test conference 2004, pp 1242–1248
34. Yen CH, Wu BF (2006) Simple error detection methods for hardware implementation of advanced encryption standard. IEEE Trans Comput 55(6):720–731
35. Zhang X, Parhi KK (2004) High-speed VLSI architectures for the AES algorithm. IEEE Trans VLSI Syst 12(9):957–967
36. Zimmermann R, Fichtner W (1997) Low-power logic styles: CMOS versus pass-transistor logic. IEEE J Solid-State Circuits 32(7):1079–1090

Mehran Mozaffari-Kermani received the B.Sc. degree from the University of Tehran in 2005, and the M.E.Sc. degree from the University of Western Ontario in 2007, in electrical engineering and computer engineering, respectively. He is currently a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Western Ontario. His current research interests include secure cryptographic systems, fault diagnosis and tolerance, VLSI reliability, and computer arithmetic.

Arash Reyhani-Masoleh received the B.Sc. degree from Iran University of Science and Technology in 1989, the M.Sc. degree from the University of Tehran in 1991, both with the first rank in electrical and electronic engineering, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo in 2001. From 1991 to 1997, he was with the Department of Electrical Engineering, Iran University of Science and Technology. From June 2001 to September 2004, he was with the Centre for Applied Cryptographic Research, University of Waterloo. In October 2004, he joined the Department of Electrical and Computer Engineering, University of Western Ontario, London, Ontario, Canada, as an Assistant Professor. His current research interests include algorithms and VLSI architectures for computations in finite fields, fault tolerant computing, and error-control coding. Dr. Reyhani-Masoleh was awarded an NSERC (Natural Sciences and Engineering Research Council of Canada) post-doctoral fellowship in 2002. Currently, he is an Associate Editor of Integration, the VLSI Journal (Elsevier).