A Lightweight Concurrent Fault Detection Scheme for the AES S-Boxes Using Normal Basis

Mehran Mozaffari-Kermani and Arash Reyhani-Masoleh

Department of Electrical and Computer Engineering,
The University of Western Ontario
London, Ontario, Canada
{mmozaff,areyhani}@uwo.ca

Abstract. The use of an appropriate fault detection scheme for hardware implementation of the Advanced Encryption Standard (AES) makes the standard robust to the internal defects and fault attacks. To minimize the overhead cost of the fault detection AES structure, we present a lightweight concurrent fault detection scheme for the composite field realization of the S-box using normal basis. The structure of the S-box is divided into blocks and the predicted parities of these blocks are obtained. Through an exhaustive search among all available composite fields and transformation matrices that map the polynomial basis representation in binary field to the normal basis representation in composite field, we have found the optimum solution for the least overhead S-box and its parity predictions. Finally, using FPGA implementations, the complexities of the proposed schemes are compared to those of the previously reported ones. It is shown that the FPGA implementations of the S-box using normal basis representation in composite fields outperform the traditional ones using polynomial basis for both with and without fault detection capability.

Keywords: Advanced encryption standard, fault detection, normal basis, S-box.

1 Introduction

The AES was approved by NIST in 2001 [1] and is currently replaced the previous Data Encryption Standard in many applications. In encryption, the AES accepts a 128-bit plaintext and a key as the inputs, where the key size can be selected as 128, 192 or 256 bits. In the AES-128, which is hereafter referred to as AES, the ciphertext is generated after 10 rounds, where each encryption round (except for the final round) consists of four transformations [1].

Among the four transformations in the encryption of the AES, only the S-box operation is non-linear. There exist several fault detection schemes devised for detecting the faults in the hardware implementation of this operation, see for example [2], [3], [4], [5], [6], [7], and [8]. In this regard, the schemes in [2], [4], [5], [6], and [7] are independent of the way this transformation is implemented

E. Oswald and P. Rohatgi (Eds.): CHES 2008, LNCS 5154, pp. 113–129, 2008.

in hardware. In [2], redundant unit is proposed for the fault detection of the S-box, where an inverse S-box is placed after the S-box. Although such an scheme detects any faults in the S-box or the inverse S-box, its overhead is at least 100%. The approach in [4] and [5] is based on storing the one-bit predicted parity of the S-box in a table and comparing it with the actual parity. Theoretically, this causes the error coverage of 50% for a single S-box. In [6], a multiplication approach for the fault detection of the multiplicative inversion of the S-box is presented. In this approach, the result of the multiplication of the input and the output of the multiplicative inversion is compared to the actual result.

There exist some other fault detection schemes that are suitable for a specific implementation of the S-box, see for example [3] and [8]. The fault detection approach presented in [3] is based on the table look-up S-boxes which may not be preferable for high performance implementations. Therefore, for applications requiring high performance AES implementations, the S-box is implemented using logic gates in the composite field [9]. This reduces the area complexity of the implementations. In addition, through pipelining, the working frequencies of the hardware implementations can be increased [10].

Since direct calculation of the multiplicative inversion is costly [9], [11], composite field arithmetic is used to perform a low cost inversion. It is noted that the inversion in $GF(2^8)$ can be implemented by mapping the binary field to the composite field using polynomial or normal bases. It is shown in [9] that the S-box structure using normal basis in composite field requires lower gate count as compared to its counterparts using polynomial basis. In this paper, we have implemented both types of the S-boxes on FPGAs and have shown that the ones using normal basis has lower complexities than the one using polynomial basis. Furthermore, we show that the parity predictions for the proposed fault detection scheme using normal basis has lower gate count and time complexity in comparison with those presented in [8] which uses polynomial basis.

In this paper, we propose a lightweight concurrent parity-based fault detection scheme for the S-box using normal basis. This scheme can also be applied to the inverse S-box. Through an exhaustive search, we obtain the least area and delay overhead S-box and its fault detection scheme for the optimum composite field. In this regard, our comparisons through FPGA implementations show that the presented scheme is more efficient than the previously reported ones. Furthermore, considering random fault injection, high error coverage is achieved for the presented scheme.

The organization of this paper is as follows: In Section 2, preliminaries regarding the AES S-box and its implementation using composite fields and normal basis are explained. The proposed fault detection scheme for the S-box is presented in Section 3. Moreover, in this section, the time and space complexities of the proposed scheme are analyzed. In Section 4, the presented fault detection scheme for the S-box and the previously reported ones are implemented on FPGAs and they are compared in terms of time and space complexities. Finally, conclusions are made in Section 5.

2 Preliminaries

In this section, we first describe the S-box operation. Then, the composite field realization of the S-box using normal basis is explained.

2.1 The S-Box

The S-box is a nonlinear operation which takes an 8-bit input and generates an 8-bit output. In the S-box, the irreducible polynomial of $P(x) = x^8 + x^4 + x^3 + x + 1$ is used to construct the binary field $GF(2^8)$. Let $X \in GF(2^8)$ and $Y \in GF(2^8)$ be the input and the output of the S-box, respectively. Then, the S-box consists of the multiplicative inversion, i.e., $X^{-1} \in GF(2^8)$, followed by an affine transformation. The affine transformation consists of the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ to generate the output as

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^{-1} + \boldsymbol{b} = \begin{pmatrix} 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{pmatrix} \boldsymbol{x}^{-1} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \tag{1}$$

where, $\boldsymbol{y}$ and $\boldsymbol{x}^{-1}$ are vectors corresponding to the field elements Y and X^{-1}, respectively.

In the following, we explain the composite field realization of the multiplicative inversion using normal basis. Then, in the next section, we propose the parity-based fault detection scheme of the S-box using this realization.

2.2 Multiplicative Inversion Using Composite Fields and Normal Basis

Let us briefly explain the composite field arithmetic to calculate the multiplicative inversion over $GF(2^8)$. In what follows, we use capital Roman letters such as X and Y for the elements in the binary field $GF(2^8)$. Furthermore, small Greek letters such as η_h, η_l, ν represent the elements in $GF(2^4)$. Finally, capital Greek letters such as Φ and Ω are utilized for the elements in $GF(2^2)$.

The transformation matrix $\boldsymbol{\Psi}$ transforms a field element $X = \sum_{i=0}^{7} x_i \alpha^i$ in the binary field $GF(2^8)$ to the corresponding representation in the composite field $GF(2^8)/GF(2^4)$. The result of this transformation is the polynomial $X = \eta_h u^{16} + \eta_l u$ (see Fig. 1), with the multiplications modulo the irreducible polynomial $u^2 + \tau u + \nu$. It is noted that the coefficients η_h and η_l are field elements in the sub-field $GF(2^4)$ representing X in terms of the normal basis $[U^{16}, U]$ [12]. The decomposition can be further applied to represent $GF(2^4)$ as a linear polynomial over $GF(2^2)$ with multiplications modulo the irreducible polynomial of $v^2 + \Omega v + \Phi$ which uses the normal basis $[V^4, V]$. Moreover, one can

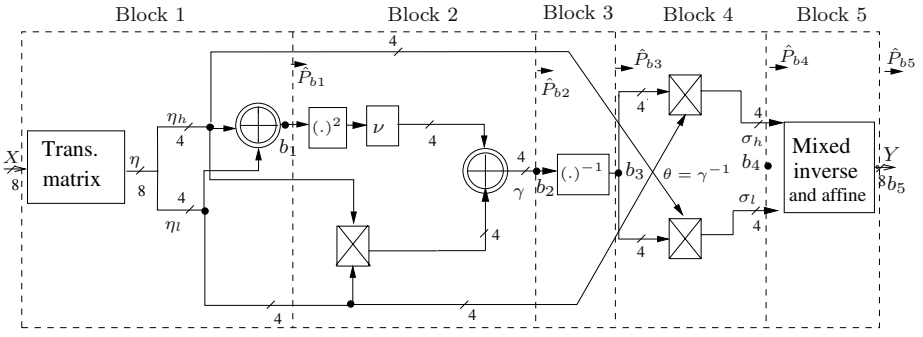

Fig. 1. The S-box using composite field and normal basis [12] and its fault detection blocks

represent $GF(2^2)$ as a linear polynomial over $GF(2)$ with multiplications modulo the irreducible polynomial of $w^2 + w + 1$ using the normal basis $[W^2, W]$. After calculating the inversion in the composite field, the inverse transformation matrix $\boldsymbol{\Psi^{-1}}$ is used to transform the composite field representation to the field element $Y = \sum_{i=0}^{7} y_i \alpha^i$ in $GF(2^8)$.

For calculating the multiplicative inversion, the most efficient choice is to let $\Omega = \tau = 1$ in the above irreducible polynomials [9], [12]. Then, we have the following for the multiplicative inversion using normal basis [12]

$$(\eta_h u^{16} + \eta_l u)^{-1} = [\theta \eta_h] u^{16} + [\theta \eta_l] u, \tag{2}$$

where, $\theta = (\eta_h \eta_l + (\eta_h{}^2 + \eta_l{}^2)\nu)^{-1}$ (see the output of block 3 in Fig. 1). As seen in (2), the multiplicative inversion consists of a number of multiplications, an inversion, a squaring and modulo-2 additions in $GF(2^4)$. In the next section, while we derive the parity predictions for the S-boxes, we will explain these in more details. Then, we derive the most efficient coefficients ν and Φ for the presented fault detection scheme.

3 Fault Detection Scheme

In this paper, we use multiple stuck-at fault model at the logic level. This type of fault, which forces multiple nodes to be stuck at logic one (for stuck-at one) or zero (for stuck-at zero) independent of the fault-free logic values, has been frequently used in the literature, see for example [16]. It is noted that the presented scheme is independent of the life time of the faults. Thus, both permanent and transient stuck-at faults lead to the same fault coverage.

In the parity-based fault detection scheme of a block of logic gates, the parity of the block is predicted and it is compared to the actual parity. The result of this comparison is the error indication flag of the corresponding block. This method has been utilized in the literature to develop a fault detection scheme for different applications, see for example [3], [13], [14], [15].

We have divided the S-box into 5 blocks similar to what is done in [8]. This results in low overhead parity predictions while maintaining the fault detection required for the security-constrained environments. This is shown in Fig. 1. Let b_i be the output of block i in Fig. 1, where $b_1 = \eta_h + \eta_l, b_2 = \gamma, b_3 = \theta, b_4 = \sigma$, and $b_5 = Y$. As seen in this figure, the first block consists of the transformation matrix that changes an element in polynomial basis to the composite field. Moreover, the last block (block 5) is obtained by mixing the inverse and affine transformation matrices. The remaining three blocks are for the multiplicative inversion, where, the hardware realization of equation (2) has been depicted. In this figure, the modulo-2 additions, consisting of 4 XOR gates, are shown by two concentric circles with a plus inside. Furthermore, the multiplications in $GF(2^4)$ are shown by rectangles with crosses inside. In the remaining of this section, the five predicted parities of the outputs of five blocks of the S-box (b_1-b_5 in Fig. 1) are obtained. The predicted parity of the output of block i is a Boolean function of the inputs of block i. These parity predictions are denoted by $\hat{P}_{b1}$, $\hat{P}_{b2}$, $\hat{P}_{b3}$, $\hat{P}_{b4}$, and $\hat{P}_{b5}$ in Fig. 1. It is noted that we have exhaustively searched for the best possible choice of ν and Φ to find the least overhead parity predictions using composite field and normal basis, the details of which are to follow.

3.1 Least Overhead Parity Predictions

The implementation complexities of different blocks of the S-box are dependent on the choice of the coefficients $\nu \in GF(2^4)$ and $\Phi \in GF(2^2)$ in the irreducible polynomials $u^2 + u + \nu$ and $v^2 + v + \Phi$ used for the composite field. Therefore, the area/delay complexities of the predicted parities of these blocks are also affected for different choices of ν and Φ. It is noted that only the values of these coefficients that make the polynomials irreducible are acceptable. Therefore, it can be derived that the only two acceptable values for Φ are $\Phi = w^2 = \{10\}_2$ and $\Phi = w = \{01\}_2$. Furthermore, among the 16 values for ν, the following 8 make the polynomial of $u^2 + u + \nu$ irreducible and thus are possible: $\nu \in \{\{\Phi 00\}_2, \{00\Phi\}_2, \{\Phi^2 00\}_2, \{00\Phi^2\}_2, \{\Phi 11\}_2, \{11\Phi\}_2, \{\Phi^2 11\}_2, \{11\Phi^2\}_2\} = \{\{0100\}_2, \{0001\}_2, \{1000\}_2, \{0010\}_2, \{0111\}_2, \{1101\}_2, \{1011\}_2, \{1110\}_2\}$.

In what follows, we are going to compare different implementations of the predicted parities of the blocks in the S-box considering different combinations of ν and Φ to reach a low complexity fault detection scheme.

Blocks 1 and 5 of the S-box. Based on the possible values of ν and Φ, the transformation matrices in blocks 1 and 5 of the S-box, denoted as $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}^{-1}$/affine, can be constructed using the algorithm presented in [17] with a slight modification for normal basis. One possible way to find the least complex transformation matrices is to calculate the Hamming weights, i.e., the number of non-zero elements, of the matrices $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}^{-1}$/affine. It is noted in [9] that instead of considering the Hamming weights, subexpression sharing is used for obtaining the low complexity implementations. We have exhaustively searched for the least overhead transformation matrices and their parity predictions combined, the results of which are presented in Table 1. In this table, for every

Table 1. Area/delay complexities of blocks 1 and 5 of the S-box and their predicted parities for possible values of νs and Φs

Φ	ν	H(Ψ)+H (Ψ^{-1}/affine)	Total area of blocks 1 and 5	Total delay of blocks 1 and 5	Total area of $\hat{P}_{b1}$ and $\hat{P}_{b5}$	Total delay of $\hat{P}_{b1}$ and $\hat{P}_{b5}$
10	0001	57	28X	$7T_X$	5X	$4T_X$
	0010	57	32X		5X	
	0100	57	34X		5X	
	1000	57	30X		5X	
	0111	67	34X		3X	
	1011	65	30X		5X	
	1101	67	34X		3X	
	1110	65	31X		5X	
01	0001	57	32X		5X	
	0010	57	32X		5X	
	0100	57	29X		5X	
	1000	57	34X		5X	
	0111	65	34X		5X	
	1011	67	37X		3X	
	1101	65	34X		5X	
	1110	67	32X		3X	

$X = XOR$, T_X= Delay of an XOR.

possible combination of ν and Φ, the Hamming weights of $\boldsymbol{\Psi}$ and $\boldsymbol{\Psi}^{-1}$/affine for the least complex cases are tabulated in column 3. Also, the number of gates needed for the low complexity implementation of blocks 1 and 5 are presented in column 4 of the table. Furthermore, the total number of XOR gates needed for the predicted parities of blocks 1 and 5 of the S-box, i.e., $\hat{P}_{b1}$ and $\hat{P}_{b5}$, and the delays associated with them are also shown in the table.

Block 2: As shown in Fig. 1, block 2 of the S-box consists of a multiplication, an addition, a squaring and a multiplication by constant ν in $GF(2^4)$. The multiplication in $GF(2^4)$ presented in [12], is depicted in Fig. 2a. As seen in this figure, it consists of three multiplications, additions and a multiplication by constant Φ in $GF(2^2)$. Moreover, the multiplication in $GF(2^2)$ is shown in this figure. The following lemmas are used for deriving the predicted parity of the multiplication in $GF(2^4)$ and block 2, respectively. It is noted that all proofs are presented in Appendix A.

Lemma 1. *Let $\lambda = (\lambda_3, \lambda_2, \lambda_1, \lambda_0)$ and $\delta = (\delta_3, \delta_2, \delta_1, \delta_0)$ be the inputs of a multiplier in $GF(2^4)$. The predicted parity of the result of the multiplication of λ and δ in $GF(2^4)$ is independent of Φ and can be derived as*

$$\hat{P}_\pi = \lambda_3\delta_3 + \lambda_2\delta_2 + \lambda_1\delta_1 + \lambda_0\delta_0. \tag{3}$$

Lemma 2. *The predicted parity of block 2, i.e., $\hat{P}_{b2}$ in Fig. 1, depends on the choice of the coefficients $\nu \in GF(2^4)$ and $\Phi \in GF(2^2)$ in the irreducible polynomials $u^2 + u + \nu$ and $v^2 + v + \Phi$ used for the composite field.*

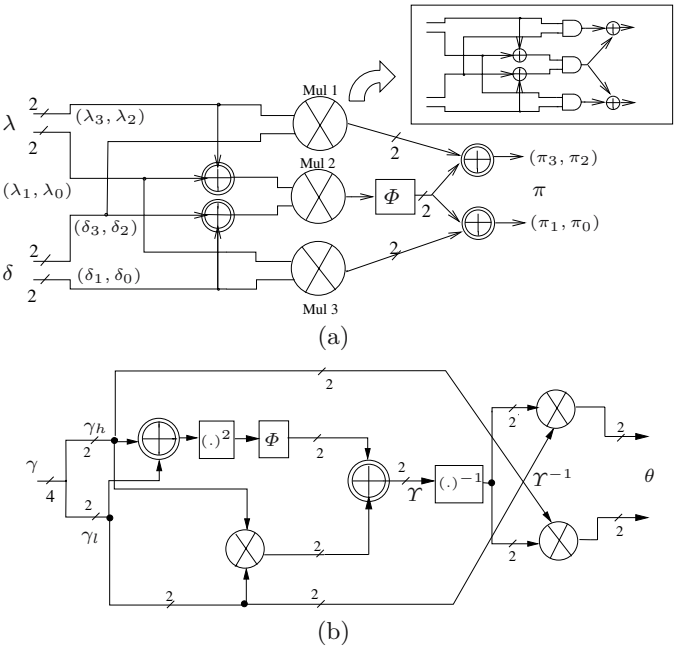

Fig. 2. (a) Multiplication and (b) Inversion in $GF(2^4)$ [12]

The proof is presented in Appendix A.

Using Lemma 1 and Lemma 2 and Fig. 1, we can state the following to predict the parity of block 2. The proof is presented in Appendix A.

Lemma 3. *The predicted parity of block 2, i.e., $\hat{P}_{b2}$, can be derived as shown in Table 2.*

Table 2 shows the predicted parities for different combinations of ν and Φ and their area/delay complexities. Moreover, the complexities for block 2 are shown in this table. As seen in Table 2, the delay overhead for both the original block and its parity prediction is the same for all the cases. Whereas, the area in terms of the number of gates are different for different values of ν and Φ.

Block 3: Block 3 in Fig. 1 consists of an inversion in $GF(2^4)$. The hardware implementation for this block has also been shown in Fig. 2b [12]. As seen in this figure, the inversion in $GF(2^4)$ is dependent on the two possible choices of Φ and is the same for different values of ν. Therefore, depending on the choice of Φ, there are two possible choices for this block and its parity prediction. It is noted that for both of these implementations, the area and the critical path delay are the same. The following theorem is used for obtaining the predicted parity of block 3, i.e., $\hat{P}_{b3}$, the proof of which is presented in Appendix A.

Table 2. Parity predictions and complexities of block 2 of the S-box in Fig. 1 for possible values of ν and Φ

Φ	ν	Area of block 2	Delay of block 2	Predicted parity ($\hat{P}_{b2}$)	Area of $\hat{P}_{b2}$	Delay of $\hat{P}_{b2}$
10	0001	28X+9A	$6T_X$ $+1T_A$	$(\eta_7 \vee \eta_3) + (\eta_6 \vee \eta_2) + (\eta_4 \vee \eta_0) + \eta_5\eta_1$	3X+3O+1A	$2T_X$ $+1T_A$
	0010	29X+9A		$(\eta_7 \vee \eta_3) + (\eta_5 \vee \eta_1) + (\eta_4 \vee \eta_0) + \eta_6\eta_2$	3X+3O+1A	
	0100	28X+9A		$(\eta_6 \vee \eta_2) + (\eta_5 \vee \eta_1) + (\eta_4 \vee \eta_0) + \eta_7\eta_3$	3X+3O+1A	
	1000	29X+9A		$(\eta_7 \vee \eta_3) + (\eta_6 \vee \eta_2) + (\eta_5 \vee \eta_1) + \eta_4\eta_0$	3X+3O+1A	
	0111	28X+9A		$(\eta_4 \vee \eta_0) + \eta_7\eta_3 + \eta_6\eta_2 + \eta_5\eta_1$	3X+3A+1O	
	1011	29X+9A		$(\eta_7 \vee \eta_3) + \eta_6\eta_2 + \eta_5\eta_1 + \eta_4\eta_0$	3X+3A+1O	
	1101	28X+9A		$(\eta_6 \vee \eta_2) + \eta_7\eta_3 + \eta_5\eta_1 + \eta_4\eta_0$	3X+3A+1O	
	1110	29X+9A		$(\eta_5 \vee \eta_1) + \eta_7\eta_3 + \eta_6\eta_2 + \eta_4\eta_0$	3X+3A+1O	
01	0001	29X+9A		$(\eta_6 \vee \eta_2) + (\eta_5 \vee \eta_1) + (\eta_4 \vee \eta_0) + \eta_7\eta_3$	3X+3O+1A	
	0010	28X+9A		$(\eta_7 \vee \eta_3) + (\eta_6 \vee \eta_2) + (\eta_5 \vee \eta_1) + \eta_4\eta_0$	3X+3O+1A	
	0100	29X+9A		$(\eta_7 \vee \eta_3) + (\eta_6 \vee \eta_2) + (\eta_4 \vee \eta_0) + \eta_5\eta_1$	3X+3O+1A	
	1000	28X+9A		$(\eta_7 \vee \eta_3) + (\eta_5 \vee \eta_1) + (\eta_4 \vee \eta_0) + \eta_6\eta_2$	3X+3O+1A	
	0111	29X+9A		$(\eta_6 \vee \eta_2) + \eta_7\eta_3 + \eta_5\eta_1 + \eta_4\eta_0$	3X+3A+1O	
	1011	28X+9A		$(\eta_5 \vee \eta_1) + \eta_7\eta_3 + \eta_6\eta_2 + \eta_4\eta_0$	3X+3A+1O	
	1101	29X+9A		$(\eta_4 \vee \eta_0) + \eta_7\eta_3 + \eta_6\eta_2 + \eta_5\eta_1$	3X+3A+1O	
	1110	28X+9A		$(\eta_7 \vee \eta_3) + \eta_6\eta_2 + \eta_5\eta_1 + \eta_4\eta_0$	3X+3A+1O	

$A = AND, \{+, X\} = XOR, \{\vee, O\} = OR$.
T_X= Delay of an XOR, T_A= Delay of an AND= Delay of an OR.

Theorem 1. *Let $\gamma = (\gamma_3, \gamma_2, \gamma_1, \gamma_0)$ be the input and $\theta = (\theta_3, \theta_2, \theta_1, \theta_0)$ be the output of an inverter in $GF(2^4)$. Then, for $\Phi = w^2 = \{10\}_2$, the predicted parity of block 3, i.e., $\hat{P}_{b3}$, can be found as*

$$\hat{P}_{b3} = \hat{P}_\theta = \overline{\gamma_2\gamma_0}(\gamma_3 + \gamma_1) + \gamma_3\gamma_1(\gamma_2 + \gamma_0). \tag{4}$$

Also, for $\Phi = w = \{01\}_2$ we have

$$\hat{P}_{b3} = \hat{P}_\theta = \overline{\gamma_3\gamma_1}(\gamma_2 + \gamma_0) + \gamma_2\gamma_0(\gamma_3 + \gamma_1). \tag{5}$$

Block 4: Block 4 of the S-box consists of two multiplications in $GF(2^4)$. According to Lemma 1, the area/delay overhead of the multiplications in $GF(2^4)$ and that of their predicted parity are the same for both $\Phi = w = \{01\}_2$ and $\Phi = w^2 = \{10\}_2$. Moreover, as seen in Fig. 1, we have $\hat{P}_{b4} = \hat{P}_{\eta_h\theta} + \hat{P}_{\eta_l\theta} = \hat{P}_{(\eta_h+\eta_l)\theta}$. Then, according to (3) in Lemma 1 with the inputs of $\eta_h + \eta_l$ and θ, one can find $\hat{P}_{b4}$ as

$$\hat{P}_{b4} = (\eta_7 + \eta_3)\theta_3 + (\eta_6 + \eta_2)\theta_2 + (\eta_5 + \eta_1)\theta_1 + (\eta_4 + \eta_0)\theta_0. \tag{6}$$

It is noted that as seen in Fig. 1, for the implementation of $\hat{P}_{b4}$, the modulo-2 additions of $\eta_7 + \eta_3$, $\eta_6 + \eta_2$, $\eta_5 + \eta_1$, and $\eta_4 + \eta_0$ are already available at the input of block 2. Therefore, implementing (6) only needs 3 XORs and 4ANDs.

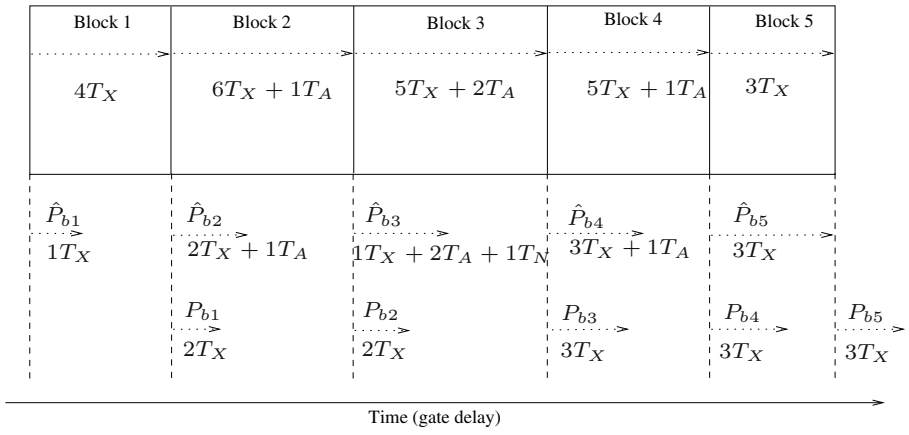

Fig. 3. Time complexity of the presented concurrent fault detection scheme for the 5 blocks of the S-box

3.2 Complexity Analysis

Based on the above discussions, the delay overhead of the predicted parities of the 5 blocks in the S-box is the same for different combinations of ν and Φ, i.e., the total delay of the predicted parities is $14T_X + 4T_A$. This delay overhead can overlap the delays for the implementations of the 5 blocks in Fig. 1. We use Fig. 3 for explaining this delay overhead in more details. As seen in this figure, the time complexity of the presented concurrent fault detection scheme for the 5 blocks of the S-box is shown. For this reason, the delays for the parity predictions, i.e., the delays for $\hat{P}_{b1}$-$\hat{P}_{b5}$, as well as the delays for the actual parity calculations[1], i.e., the delays for P_{b1}-P_{b5}, are depicted in this figure. As seen in Fig. 3, the delays for 5 parity predictions can overlap the time needed for computations in the corresponding blocks. After finding the predicted parity for a block, say block i, the actual parity of this block is obtained during the time needed for the computation of block $i + 1$. As seen in Fig. 3, the only delay overhead for this concurrent scheme is the delay of the actual parity of block 5 which is $3T_X$.

Using the discussions presented in this section and the results of Tables 1 and 2, the total gate count of all blocks of the S-box and their parity predictions for different combinations of ν and Φ are shown in Table 3. As seen in this table, if we only consider the area complexities of the parity predictions, the following four composite fields have the least area: $\Phi = \{10\}_2$ and $\nu \in \{\{0111\}_2, \{1101\}_2\}$ and also $\Phi = \{01\}_2$ and $\nu \in \{\{1011\}_2, \{1110\}_2\}$. As seen from Table 3, the gates needed for implementing these low-area parity predictions are 12 XORs, 11 ANDs, 1 OR and 1 NOT. However, the results of the table show that none of these has the least area for the S-box and its fault detection circuit together. As seen in Table 3, among all the 16 possible combinations of ν and Φ, the composite

[1] Binary trees of XOR gates are used.

Table 3. Area complexities of blocks 1 to 5 of the S-box and their predicted parities for possible values of νs and Φs

Φ	ν	Area of blocks 1 to 5	Area of $\hat{P}_{b1}$ to $\hat{P}_{b5}$	Total area of the S-box and its parity predictions
10	**0001**	**119X+36A**	**14X+9A+3O+1N**	**133X+45A+3O+1N**
	0010	124X+36A	14X+9A+3O+1N	138X+45A+3O+1N
	0100	125X+36A	14X+9A+3O+1N	139X+45A+3O+1N
	1000	122X+36A	14X+9A+3O+1N	136X+45A+3O+1N
	0111	125X+36A	12X+11A+1O+1N	137X+47A+1O+1N
	1011	122X+36A	14X+11A+1O+1N	136X+47A+1O+1N
	1101	125X+36A	12X+11A+1O+1N	137X+47A+1O+1N
	1110	123X+36A	14X+11A+1O+1N	137X+47A+1O+1N
01	0001	124X+36A	14X+9A+3O+1N	138X+45A+3O+1N
	0010	123X+36A	14X+9A+3O+1N	137X+45A+3O+1N
	0100	121X+36A	14X+9A+3O+1N	135X+45A+3O+1N
	1000	125X+36A	14X+9A+3O+1N	139X+45A+3O+1N
	0111	126X+36A	14X+11A+1O+1N	140X+47A+1O+1N
	1011	128X+36A	12X+11A+1O+1N	140X+47A+1O+1N
	1101	126X+36A	14X+11A+1O+1N	140X+47A+1O+1N
	1110	123X+36A	12X+11A+1O+1N	135X+47A+1O+1N

$X = XOR$, $A = AND$, $O = OR$, $N = NOT$.

field shown in bold face, i.e., $\Phi = w^2 = \{10\}_2$ and $\nu = \{00\Phi^2\}_2 = \{0001\}_2$, has the least area for the S-box and its fault detection circuit combined. It is interesting to note that after area optimization, this field has also been suggested in [9] for reaching the least area S-box using composite field.

From the previous section, we reach the following parity predictions for the 5 blocks in order to obtain the least overhead S-box and its fault detection circuit in Fig. 1

$$\hat{P}_{b1} = x_7 + x_5, \tag{7}$$

$$\hat{P}_{b2} = (\eta_7 \vee \eta_3) + (\eta_6 \vee \eta_2) + (\eta_4 \vee \eta_0) + \eta_5\eta_1, \tag{8}$$

$$\hat{P}_{b3} = \overline{\gamma_2\gamma_0}(\gamma_3 + \gamma_1) + \gamma_3\gamma_1(\gamma_2 + \gamma_0), \tag{9}$$

$$\hat{P}_{b4} = (\eta_7 + \eta_3)\theta_3 + (\eta_6 + \eta_2)\theta_2 + (\eta_5 + \eta_1)\theta_1 + (\eta_4 + \eta_0)\theta_0, \tag{10}$$

$$\hat{P}_{b5} = \sigma_7 + \sigma_5 + \sigma_4 + \sigma_3 + \sigma_2, \tag{11}$$

where, $\vee$ represents an OR operation.

We conclude this section with the calculation of the error coverage of the presented scheme. As seen in Fig. 1, since the 5 blocks of the S-box do not overlap, the fault detection of each block is independent of those of the others. It is noted that using parities, the probability of detecting (or not detecting) the faults by

the error indication flag of each block is $\frac{1}{2}$. Therefore, using the mentioned fault model, for the error coverage of each S-box we have $100 \times (1 - (\frac{1}{2})^5)\% = 97\%$.

4 Comparisons

In this section, we compare the area and the delay of the presented scheme with those of the previously reported ones. For this reason, first the gate count and the gate delays of the schemes are obtained and compared. Then, the implementations on the Xilinx [18] FPGAs are presented.

4.1 Area and Time Complexities

For deriving the area overhead of the presented fault detection scheme, we assume that 2-input AND and OR gates require 6 transistors each using the full CMOS technology. Also, 2-input XOR and XNOR gates can be implemented using 10 transistors each [19] and a NOT gate can be realized using 2 transistors assuming that PMOS and NMOS need the same chip area. Therefore, the space complexities of a 2-input AND (OR) and a NOT gate are equivalent to 0.6 and 0.2 XOR gates, respectively.

In the previous section, the total gate count for the predicted parities of 5 blocks of the S-box was derived as 14 XORs, 9 ANDs, 3 ORs and 1 NOT which is equivalent to 21.4 XORs. In addition, 23 XORs and 5 XORs are needed for obtaining the actual parities and the comparisons of the predicted and the actual parities, respectively. Therefore, the equivalent gates for the total area overhead is obtained as $21.4 + 28 = 49.4$ XOR gates. Moreover, as seen in Table 3, the corresponding S-box implementation needs 119 XORs and 36 ANDs which is equivalent to 140.6 XOR gates. Therefore, the percentage of area overhead is approximately $\frac{49.4}{140.6} \simeq 35\%$.

The complexity of the presented fault detection scheme can be compared to the other fault detection schemes of the S-box using composite fields. The original area of the S-boxes and the overheads of these fault detection schemes in terms of equivalent XOR gates are presented in Table 4. The S-box presented in [11] has been hardware optimized in [20] and is extensively used in the literature, see for example [21], [22]. The gate count for the predicted parities of this S-box in [20] is derived as 23 XORs, 10 ANDs and 1 XNOR [8]. In addition, similar to the scheme in this paper, 28 XORs are required for obtaining the actual parities and their comparisons with the predicted parities. Therefore, the equivalent gates for the total area overhead is obtained as 58 XORs. Moreover, this S-box needs 123 XORs and 36 ANDs, equivalent to 144.6 XOR gates, resulting in the area overhead of approximately 40%. In addition, for comparison, we have derived the parity predictions of the S-box using polynomial basis, presented in [23] which is used in the literature, see for example [24], [25], [26]. Table 4 shows the total equivalent gate count for the fault detection scheme of this S-box, comprising the actual and the predicted parities and comparisons. As seen in this table, the gate overhead for the fault detection scheme of this S-box is around 38%.

Table 4. Area overhead comparison of the parity-based fault detection schemes

S-box	[11], [20], [21], [22]	[23], [24], [25], [26]	[27]	**Presented**
FDS[a]	[8]	Applied[b]	Applied[b]	**This work**
S-box XORs	144.6	144.6	141.6	**140.6**
FDS XORs[c]	58	56	56.4	**49.4**
Area overhead	40%	38%	39%	**35%**

[a] Fault detection scheme.

[b] We have applied the technique proposed in [8] to derive the predicted parities for the 5 blocks of the S-boxes presented in [23]-[27].

[c] The area complexity overhead of the fault detection scheme is dependent on the way the S-box is implemented. Therefore, the numbers of XOR gates are different for different S-box realizations.

Finally, using subexpression sharing for the implementation of the S-box in [27], the area overhead of the fault detection scheme is approximately 39% (see Table 4). As seen in this table, all the above S-boxes and their fault detection S-boxes are less compact than the scheme presented in this paper. It is noted that similar to the presented fault detection scheme, the delay overhead of these schemes is $4T_X$ comprising $3T_X$ for calculating the actual parity of block 5 plus one T_X for its comparison with the predicted parity of this block.

4.2 FPGA Implementations

In the following, we have implemented the S-boxes using look-up table (LUT) and the ones presented in [20], [23], and [27] which use polynomial basis (PB) representation in composite field. We have also implemented the fault detection schemes proposed in [2] and [3] which are based on the LUT implementation of the S-box and the one presented in [8] which is based on the S-box of [20]. Moreover, we have applied similar technique presented in [8] and derived formulations for the S-boxes of [23] and [27] to implement their fault detection schemes. All the schemes are implemented on the Xilinx Virtex^{TM}-E and Virtex^{TM}-II Pro FPGAs [18] and are compared with the one presented in this paper. For the implementations, VHDL has been used as the design-entry language for the Xilinx ISE^{TM} version 9.1i. Furthermore, the synthesis is performed using XST^{TM}.

For the presented scheme in this paper, we have implemented the S-box presented in the previous section and the fault detection circuits, i.e., the equations (7)-(11). The results of the implementations have been tabulated in Table 5. In this table, the synthesis optimization goal is set as area with medium effort. The number of slices used for the implementations on the target devices and the minimum clock periods (delays) are presented in this table. It is noted that we have not used sub-pipelining in the implementations of the S-boxes using composite fields since the results are intended for finding the space complexity of the S-boxes and the overhead of their fault detection schemes.

Table 5. Comparisons of the implementation of the fault detection scheme of the S-box using normal basis (NB) with those of other schemes on Xilinx FPGAs

FPGA family (Device)	S-box		Slice			Delay (ns)		
	Structure	FDS	Original	FDS	Overhead[a]	Original	FDS	Overhead[a]
Virtex-E (xcv50e-8)	LUT	[2]	88	188	113.6%	6.280	8.621	37.3%
	LUT	[3]	88	206	134.1%[b]	6.280	8.242	31.2%
	PB [20]	[8]	33	42	27.3%	17.976	19.079	6.1%
	PB [23]	Applied[c]	38	50	31.6%	15.875	17.077	7.8%
	PB [27]	Applied[c]	37	47	27.0%	19.133	19.912	4.1%
	NB	**This work**	**31**	**39**	**25.8%**	**16.517**	**17.360**	**5.1%**
Virtex-2 Pro (xc2vp2-7)	LUT	[2]	69	150	117.4%	3.826	5.398	41.0%
	LUT	[3]	69	159	130.4%[b]	3.826	4.287	12.0%
	PB [20]	[8]	33	42	27.3%	9.375	10.317	10.0%
	PB [23]	Applied[c]	38	50	31.6%	8.285	9.582	15.7%
	PB [27]	Applied[c]	37	47	27.0%	9.986	10.832	8.4%
	NB	**This work**	**31**	**39**	**25.8%**	**9.339**	**10.026**	**9.8%**

[a] Overhead=$\frac{FDS-original}{original} \times 100$.

[b] The high area overhead is because of using two blocks of 256×9 memory cells to generate the predicted parity bit and the 8-bit output of the S-box [3].

[c] We have applied the technique proposed in [8] to derive the predicted parities for the 5 blocks of the S-boxes presented in [23] and [27].

The results of the comparison of the presented fault detection scheme for the S-box using normal basis with those of the other schemes have also been presented in Table 5. Using pipelined distributed memories, we have implemented the fault detection scheme presented in [2], which is based on using redundant units for the S-box of table look-ups. Furthermore, the fault detection scheme proposed in [3] is implemented. This scheme uses 512×9 memory cells to generate the predicted parity bit and the 8-bit output of the S-box [3]. The results in Table 5 show that for both of these schemes the area overhead is more than 100%. We have also implemented the fault detection scheme presented in [8] which uses the original S-box proposed in [20]. This S-box uses the polynomial basis representation in composite field. Our implementations show that the implementation of [20] is more area-efficient on FPGAs than the ones presented in [23] and [27]. However, the one presented in [23] is the fastest one compared to [20] and [27]. Moreover, we have applied the fault detection schemes presented in [8] to the S-boxes in [23] and [27]. Those fault detection schemes have also been implemented and the implementation results are also shown in this table.

As seen in bold faces in the table, the presented S-box is the most compact one among the other S-boxes with and without the fault detection scheme. It is interesting to note that the least area required for the S-box implementation using normal basis on FPGAs complies with the least gate count reported in [12] for such a composite field. Moreover, except for the scheme in [27], the post place and route timing overhead of the presented scheme is less than the other schemes. It is interesting to note that the presented scheme has less delay for the

fault detection S-box compared to the scheme for [27]. This delay can overlap the computation time of the next transformation in the AES encryption.

5 Conclusions

In this paper, we have presented a high performance parity-based concurrent fault detection scheme of the S-box using normal basis for the advanced encryption standard. We have exhaustively searched for the least complex S-box as well as its fault detection circuit and have presented closed formulations for the parity predictions of each block of the S-box. We have implemented a number of proposed S-boxes and their fault detection schemes from the literature on FPGAs and compared them with the one presented here. Our FPGA implementations using area optimized syntheses show that the S-box using normal basis is more compact than the one using polynomial basis. Moreover, the cost of the FPGA implementation of the presented fault detection scheme is 25.8% slice overhead with negligible timing delay. It is noted that similar parity-based fault detection scheme can be obtained for the inverse S-box in the AES decryption.

Acknowledgments

The authors would like to thank the reviewers of CHES 2008 for their comments. This work has been supported by an NSERC Discovery grant awarded to A. Reyhani-Masoleh.

References

1. Federal Information Processing Standards Publication 197, Advanced Encryption Standard (AES), NIST (2001), `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`
2. Karri, R., Wu, K., Mishra, P., Yongkook, K.: Fault-based Side-Channel Cryptanalysis Tolerant Rijndael Symmetric Block Cipher Architecture. In: DFT 2001, pp. 418–426. IEEE Computer Society Press, San Francisco (2001)
3. Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., Piuri, V.: Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard. IEEE Trans. on Computers, special issue on Cryptographic Hardware and Embedded Systems 52(4), 492–505 (2003)
4. Bertoni, G., Breveglieri, L., Koren, I.: An Efficient Hardware-based Fault Diagnosis Scheme for AES: Performances and Cost. In: DFT 2004, pp. 130–138. IEEE Computer Society Press, Cannes (2004)
5. Breveglieri, L., Koren, I.: Maistri: Incorporating Error Detection and Online Reconfiguration into a Regular Architecture for the Advanced Encryption Standard. In: DFT 2005, pp. 72–80. IEEE Computer Society Press, Monterey (2005)
6. Karpovsky, M., Kulikowski, K.J., Taubin, A.: Differential Fault Analysis Attack Resistant Architectures for the Advanced Encryption Standard. In: Quisquater, J.J., Paradinas, P., Deswarte, Y., Abou El Kalam, A. (eds.) CARDIS 2004, vol. 153, pp. 177–192. Kluwer, Toulouse (2004)

7. Yen, C.H., Wu, B.F.: Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard. IEEE Trans. on Computers 55(6), 720–731 (2006)
8. Mozaffari-Kermani, M., Reyhani-Masoleh, A.: Parity-based Fault Detection Architecture of S-box for Advanced Encryption Standard. In: DFT 2006, pp. 572–580. IEEE Computer Society Press, Arlington (2006)
9. Canright, D.: A Very Compact Rijndael S-box. Technical Report: NPS-MA-05-001, Naval Postgraduate School (2005)
10. Hodjat, A., Verbauwhede, I.: Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s AES Processors. IEEE Trans. on Computers 55(4), 366–372 (2006)
11. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
12. Canright, D.: A Very Compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
13. Cardarilli, G.C., Ottavi, M., Pontarelli, S., Re, M., Salsano, A.: Fault Localization, Error Correction, and Graceful Degradation in Radix 2 Signed Digit-based Adders. IEEE Trans. on Computers 55(5), 534–540 (2006)
14. Cardarilli, G.C., Pontarelli, S., Re, M., Salsano, A.: A Self Checking Reed Solomon Encoder: Design and Analysis. In: DFT 2005, pp. 111–119. IEEE Computer Society Press, Monterey (2005)
15. Nicolaidis, M.: Carry Checking/Parity Prediction Adders and ALUs. IEEE Trans. on VLSI Systems. 11(1), 121–128 (2003)
16. Monnet, Y., Renaudin, M., Leveugle, R.: Designing Resistant Circuits against Malicious Faults Injection Using Asynchronous Logic. IEEE Trans. on Computers 55(9), 1104–1115 (2006)
17. Zhang, X., Parhi, K.K.: On the Optimum Constructions of Composite Field for the AES Algorithm. IEEE Trans. on Circuits and Systems II: Express Briefs 53(10), 1153–1157 (2006)
18. Xilinx, `http://www.xilinx.com/`
19. Zimmermann, R., Fichtner, W.: Low-power Logic Styles: CMOS versus Pass-transistor Logic. IEEE Journal of Solid-State Circuits 32(7), 1079–1090 (1997)
20. Zhang, X., Parhi, K.K.: High-Speed VLSI Architectures for the AES Algorithm. IEEE Trans. on VLSI Systems. 12(9), 957–967 (2004)
21. Morioka, S., Satoh, A.: An Optimized S-Box Circuit Architecture for Low Power AES Design. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 172–186. Springer, Heidelberg (2003)
22. Standaert, F.X., Rouvroy, G., Quisquater, J.J., Legat, J.D.: Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs. In: D.Walter, C., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 334–350. Springer, Heidelberg (2003)
23. Wolkerstorfer, J., Oswald, E., Lamberger, M.: An ASIC Implementation of the AES SBoxes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 67–78. Springer, Heidelberg (2002)
24. Trichina, E.: Combinational Logic Design for AES Subbyte Transformation on Masked Data. In: Cryptology eprint archive: Report 2003/236, IACR, Report 2003/236 (2003), `http://eprint.iacr.org/`
25. Mangard, S., Aigner, M., Dominikus, S.: A Highly Regular and Scalable AES Hardware Architecture. IEEE Trans. on Computers. 52(4), 483–491 (2003)

26. Mangard, S., Pramstaller, N., Oswald, E.: Successfully Attacking Masked AES Hardware Implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005)
27. Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-Box. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 323–333. Springer, Heidelberg (2005)

Appendix A: Proofs

Proof of Lemma 1. According to Fig. 2a, for the inputs $\Lambda = (\Lambda_1, \Lambda_0)$ and $\Delta = (\Delta_1, \Delta_0)$, the two-bit result of the multiplication in $GF(2^2)$, $\Pi = (\Pi_1, \Pi_0)$, can be derived as $\Pi_1 = \Delta_1\Lambda_0 + \Delta_0\Lambda_1 + \Delta_0\Lambda_0$ and $\Pi_0 = \Delta_1\Lambda_0 + \Delta_0\Lambda_1 + \Delta_1\Lambda_1$. Furthermore, multiplication by two possible values of Φ, i.e., $\Phi = w^2 = \{10\}_2$ and $\Phi = w = \{01\}_2$, can be obtained by putting $\Delta = \Phi$. Then, we have $\Pi_1 = \Lambda_0$ and $\Pi_0 = \Lambda_1 + \Lambda_0$ for $\Phi = w^2 = \{10\}_2$ and $\Pi_1 = \Lambda_1 + \Lambda_0$ and $\Pi_0 = \Lambda_1$ for $\Phi = w = \{01\}_2$. Consequently, one can derive the coordinates of π according to Fig. 2a and these discussions for the operations in the multiplication $GF(2^4)$. Therefore, for $\Phi = w^2 = \{10\}_2$ we have

$$\begin{aligned}
\pi_3 &= \lambda_3(\delta_3 + \delta_1 + \delta_0) + \lambda_2(\delta_1 + \delta_2) + \lambda_1(\delta_3 + \delta_2 + \delta_1 + \delta_0) + \lambda_0(\delta_3 + \delta_1),\\
\pi_2 &= \lambda_3(\delta_2 + \delta_1) + \lambda_2(\delta_3 + \delta_2 + \delta_0) + \lambda_1(\delta_3 + \delta_1) + \lambda_0(\delta_2 + \delta_0),\\
\pi_1 &= \lambda_3(\delta_3 + \delta_2 + \delta_1 + \delta_0) + \lambda_2(\delta_3 + \delta_1) + \lambda_1(\delta_3 + \delta_2 + \delta_1) + \lambda_0(\delta_3 + \delta_0),\\
\pi_0 &= \lambda_3(\delta_3 + \delta_1) + \lambda_2(\delta_2 + \delta_0) + \lambda_1(\delta_3 + \delta_0) + \lambda_0(\delta_2 + \delta_1 + \delta_0).
\end{aligned} \tag{12}$$

Also, for $\Phi = w = \{01\}_2$ we have the result as

$$\begin{aligned}
\pi_3 &= \lambda_3(\delta_3 + \delta_2 + \delta_1) + \lambda_2(\delta_3 + \delta_0) + \lambda_1(\delta_3 + \delta_1) + \lambda_0(\delta_2 + \delta_0),\\
\pi_2 &= \lambda_3(\delta_3 + \delta_0) + \lambda_2(\delta_2 + \delta_1 + \delta_0) + \lambda_1(\delta_2 + \delta_0) + \lambda_0(\delta_3 + \delta_2 + \delta_1 + \delta_0),\\
\pi_1 &= \lambda_3(\delta_3 + \delta_1) + \lambda_2(\delta_2 + \delta_0) + \lambda_1(\delta_3 + \delta_1 + \delta_0) + \lambda_0(\delta_2 + \delta_1),\\
\pi_0 &= \lambda_3(\delta_2 + \delta_0) + \lambda_2(\delta_3 + \delta_2 + \delta_1 + \delta_0) + \lambda_1(\delta_2 + \delta_1) + \lambda_0(\delta_3 + \delta_2 + \delta_0).
\end{aligned} \tag{13}$$

Modulo-2 adding the coordinates of (12) or (13) gives (3) and the proof is complete. In addition, another proof can be obtained by observing Fig. 2a and noting that the output of the multiplication by Φ is added to both of the results, i.e., it is added to both (π_3, π_2) and (π_1, π_0). Therefore, it is canceled in finding the predicted parity. □

Proof of Lemma 2. Considering the fact that $\hat{P}_{b2} = \hat{P}_{(\eta_h+\eta_l)^2\nu} + \hat{P}_{\eta_h\eta_l}$, one can use Lemma 1 to obtain $\hat{P}_{\eta_h\eta_l}$ independent of the values of ν and Φ. However, $\hat{P}_{(\eta_h+\eta_l)^2\nu}$ depends on the elements ν and Φ. This is because of having squaring in $GF(2^4)$, i.e., $(\eta_h + \eta_l)^2$, and also a multiplication by ν to obtain $\hat{P}_{(\eta_h+\eta_l)^2\nu}$. Therefore, the predicted parity of block 2 is also dependent on these values and the proof is complete. □

Proof of Lemma 3. One can use Lemma 1 to obtain $\hat{P}_{(\eta_h+\eta_l)^2\nu}$ and $\hat{P}_{\eta_h\eta_l}$ in $\hat{P}_{b2} = \hat{P}_{(\eta_h+\eta_l)^2\nu} + \hat{P}_{\eta_h\eta_l}$. $\hat{P}_{\eta_h\eta_l}$ can be easily found using Lemma 1. Furthermore,

using Lemma 1 with the inputs being $\lambda = (\eta_h + \eta_l)^2$ and $\delta = \nu$ one can obtain $\hat{P}_{(\eta_h+\eta_l)^2\nu}$. Noting that the possible values for Φ are $\Phi = w^2 = \{10\}_2$ and $\Phi = w = \{01\}_2$, one can find the corresponding possible $(\eta_h + \eta_l)^2$ using (12) and (13). This is achieved by putting both inputs in (12) or (13) as $\eta_h + \eta_l$. Then, for $\Phi = w^2 = \{10\}_2$ we have

$$\begin{aligned}(\eta_h + \eta_l)^2 =&(\eta_7 + \eta_6 + \eta_5 + \eta_3 + \eta_2 + \eta_1, \eta_6 + \eta_5 + \eta_4 + \eta_2 + \eta_1 + \eta_0,\\ &\eta_7 + \eta_5 + \eta_4 + \eta_3 + \eta_1 + \eta_0, \eta_7 + \eta_6 + \eta_4 + \eta_3 + \eta_2 + \eta_0),\end{aligned} \tag{14}$$

and for $\Phi = w = \{01\}_2$ we have

$$\begin{aligned}(\eta_h + \eta_l)^2 =&(\eta_7 + \eta_5 + \eta_4 + \eta_3 + \eta_1 + \eta_0, \eta_7 + \eta_6 + \eta_4 + \eta_3 + \eta_2 + \eta_0,\\ &\eta_7 + \eta_6 + \eta_5 + \eta_3 + \eta_2 + \eta_1, \eta_6 + \eta_5 + \eta_4 + \eta_2 + \eta_1 + \eta_0).\end{aligned} \tag{15}$$

One can obtain the predicted parities of block 2, i.e., $\hat{P}_{b2} = \hat{P}_{(\eta_h+\eta_l)^2\nu} + \hat{P}_{\eta_h\eta_l}$, for all the possible combinations of ν and Φ. The results are presented in Table 2. □

Proof of Theorem 1. According to Fig. 2b, $\hat{P}_\theta = \hat{P}_{\Upsilon^{-1}\gamma_h} + \hat{P}_{\Upsilon^{-1}\gamma_l} = \hat{P}_{\Upsilon^{-1}(\gamma_h+\gamma_l)}$. Then, according to the predicted parity of the multiplication in $GF(2^2)$ in the proof of Lemma 1, we have $\hat{P}_{\Upsilon^{-1}(\gamma_h+\gamma_l)} = \Upsilon_1^{-1}(\gamma_3+\gamma_1)+\Upsilon_0^{-1}(\gamma_2+\gamma_0)$. Moreover, considering the fact that the inversion in $GF(2^2)$ is free, i.e., $\Upsilon^{-1} = (\Upsilon_0, \Upsilon_1)$, we reach $\hat{P}_\theta = \Upsilon_0(\gamma_3 + \gamma_1) + \Upsilon_1(\gamma_2 + \gamma_0)$. Then, according to the formulations for the multiplication in $GF(2^2)$ and knowing that the squaring in $GF(2^2)$ is free, finding the coordinates of Υ for two values of Φ is straightforward and the proof is complete. □