Fault Detection Architectures for Field Multiplication Using Polynomial Bases

Arash Reyhani-Masoleh[1] and M. Anwar Hasan[2]

[1] Department of Electrical and Computer Engineering

University of Western Ontario, London, Ontario, Canada N6A 5B9

[2] Department of Electrical and Computer Engineering

University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

E-mails: areyhani@eng.uwo.ca and ahasan@ece.uwaterloo.ca

Corresponding author: A. Reyhani-Masoleh

Abstract

In many cryptographic schemes, the most time consuming basic arithmetic operation is the finite field multiplication and its hardware implementation for bit parallel operation may require millions of logic gates. Some of these gates may become faulty in the field due to natural causes or malicious attacks, which may lead to the generation of erroneous outputs by the multiplier. In this paper, we propose new architectures to detect erroneous outputs caused by certain types of faults in bit-parallel and bit-serial polynomial basis multipliers over finite fields of characteristic two. In particular, parity prediction schemes are developed for detecting errors due to single and certain multiple stuck-at faults. Although the issue of detecting soft errors in registers is not considered, the proposed schemes have the advantage that they can be used with any irreducible binary polynomial chosen to define the finite field.

Key words: Finite fields, polynomial basis multiplier, error detection.

1 Introduction

Among the basic arithmetic operations over finite fields $GF(2^m)$, multiplication is the one which has received most attention in the literature [10, 6, 20, 14]. This is mainly because the implementation of a multiplier is much more complex compared to a finite field adder and using multiplication operation repeatedly one can perform other difficult field operations, such as inversion and exponentiation, which are extensively used in cryptographic systems [1, 18].

Finite field multiplication is quite different from its counterparts in integer and floating point number systems. For todays cryptographic applications, the field size can be very large and each input of the multiplier can be 160 to 2048 bits long. Such a multiplier may require millions of logic gates and it is possible that errors will occur in the computation due to faults in the field. If one can have a multiplier which is capable of detecting error on-line at the presence of certain faults, cryptographic schemes can be operated more reliably.

In an attempt to detect errors in finite field multipliers, the authors of [5] have considered bit-serial multipliers in $GF(2^m)$ and have presented error detection schemes for four types of multipliers using a parity prediction technique. Their polynomial basis scheme for error detection is applicable to a special class of fields. These fields are defined using irreducible all-one polynomials that are available for certain values of m only. Additionally, when an all-one polynomial is irreducible, the corresponding m is not prime. This makes many designers to avoid such a value of m and the corresponding irreducible all-one polynomial that defines the underlying field for certain cryptosystems, such as those based on elliptic curves.

The parity prediction technique has been used for detecting faults in cryptographic architecture of the Advanced Encryption Standard (AES), see for example [2, 19]. The AES encrypts (and decrypts) a data block using a block key and consists of eleven rounds and each round has four operations. The data block is divided into a number of bytes, e.g., 16 bytes in the 128-bit data block, and each byte is an element in the finite field $GF(2^8)$ generated by $z^8 + z^4 + z^3 + z + 1$. Then, each round of the AES algorithm uses arithmetic operations over $GF(2^8)$. In [2], the predicted parity of each byte is determined and then compared with the actual parity. In their scheme, 16 parity bits are predicted and checked in the three possible choices: (i) once at the end of each round, (ii) four times at the end of each operations in every round, (iii) once at the end of the final round [2]. The authors of [19] have proposed a low cost error detection scheme for the hardware architecture of the AES. Their scheme requires one parity bit for one 128-bit data block and the predicted parity is compared with the actual parity once in each round.

The importance of eliminating errors in cryptographic computations has been pointed out in some recent articles, for examples [3, 8, 4]. The presence of faults in cryptosystems can lead to an active attack which results in leakage of secrete information from the cryptosystem. The simplest way to prevent such an attack is to ensure that the computational device verifies the values it computes before sending them out [4].

In this paper, we consider fault detection architectures for $GF(2^m)$ multipliers of both bit-parallel and bit-serial types. The well-known polynomial basis is used for representing the field elements. We develop parity prediction schemes for detecting errors due to single and certain multiple faults during the multiplication operation in the field. These new schemes can be used for any field defining irreducible binary polynomial. It is worth mentioning that in the case of multiple faults due to malicious attacks or natural causes that result in even number of errors at the output, the proposed structures are unable to detect these faults. These structures are also not suitable for detecting soft errors.

The organization of this papers is as follows. In Section 2, some preliminaries regarding multiplication using polynomial basis and our fault detection strategy are given. Then, in Sections 3 and 4 fault detection architectures for traditional and recently proposed bit-parallel multipliers are presented. Explicit formulations for parity predictions of three irreducible polynomials, namely equally-spaced polynomials, trinomials and pentanomials, are also given in Section 4. In Section 5, similar techniques are used to develop fault detection architectures for both MSB-first and LSB-first bit-serial multipliers. Finally conclusions are made in Section 6.

2 Preliminaries

2.1 Bit-Parallel Polynomial Basis Multipliers

Let the monic irreducible binary polynomial that defines the field $GF(2^m)$ be

$$F(z) = z^m + \sum_{i=0}^{m-1} f_i z^i \tag{1}$$

of degree m, where $f_i{\in}GF(2)$ for $i = 0,\ 1,\ \cdots,\ m-1$. Let $\alpha \in GF(2^m)$ be a root of $F(z)$, *i.e.*, $F(\alpha) = 0$. Then the set $\{1,\ \alpha,\ \alpha^2,\ \cdots,\ \alpha^{m-1}\}$ is known as the polynomial (or standard) basis and each element of $GF(2^m)$ can be written with respect to (w.r.t.) this basis, *i.e.,* if A is an element of $GF(2^m)$, then

$$A = \sum_{i=0}^{m-1} a_i \alpha^i,\ a_i \in \{0,\ 1\}, \tag{2}$$

where a_is are the coordinates of A w.r.t. polynomial basis (PB). For convenience, these coordinates will be denoted in vector notation as

$$\mathbf{a} = [a_0,\ a_1,\ a_2,\ \cdots\ a_{m-1}]^T, \tag{3}$$

where T denotes the transposition of a vector.

Traditional Multiplier

Let C be the product of any two elements A and B of $GF(2^m)$. Then, C can be obtained w.r.t. the PB using two steps of polynomial multiplication and modular reduction as follows:

$$S = A \cdot B = A \cdot \sum_{i=0}^{m-1} b_i \alpha^i = \sum_{i=0}^{m-1} b_i \cdot (A\alpha^i), \tag{4}$$

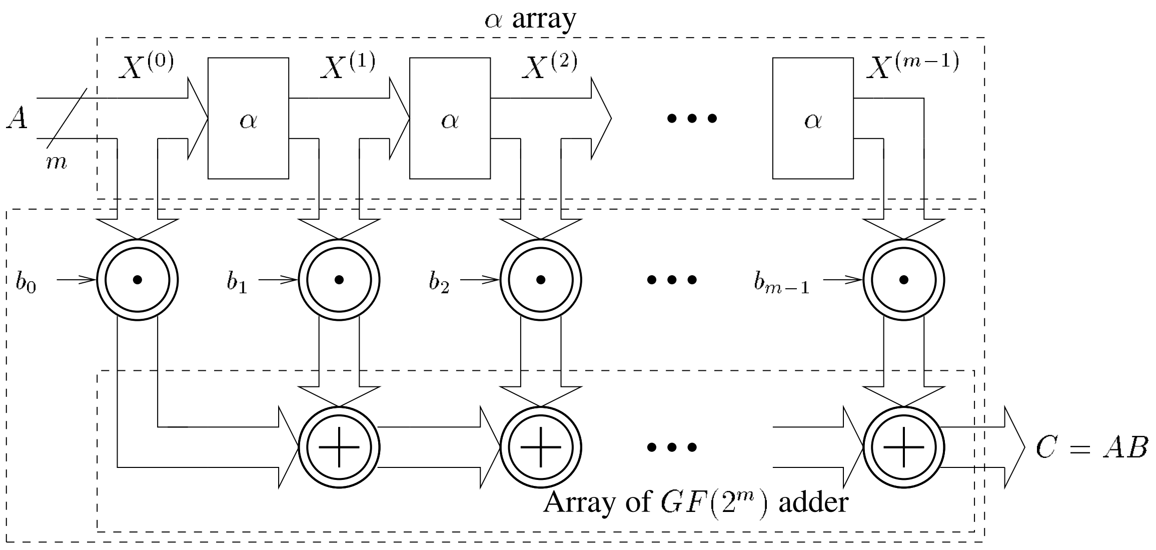


Figure 1: Multiplication of two elements in $GF(2^m)$.

$$C = S \bmod F(\alpha) = \sum_{i=0}^{m-1} b_i \cdot ((A\alpha^i) \bmod F(\alpha)) \tag{5}$$

$$= \sum_{i=0}^{m-1} b_i \cdot X^{(i)}, \tag{6}$$

where

$$X^{(i)} = \alpha \cdot X^{(i-1)} \bmod F(\alpha), \quad 1 \le i \le m-1 \tag{7}$$

and $X^{(0)} = A$.

A traditional bit-parallel architecture for $GF(2^m)$ multiplication using (6) is shown in Figure 1. It mainly consists of three types of modules, namely, sum, pass-thru and α modules. The sum module (denoted as a double circle with a plus inside) is to simply add two $GF(2^m)$ elements and it can be realized in hardware using m two-input XOR gates. The pass-thru module (denoted as a double circle with a dot inside) is to multiply a $GF(2^m)$ element by a $GF(2)$ element, *i.e.*, if $X^{(i)} \in GF(2^m)$ and $b_i \in GF(2)$ are two inputs to a pass-thru module, then its output is $b_i X^{(i)} = \begin{cases} X^{(i)} & \text{if} \quad b_i = 1, \\ 0 & \text{if} \quad b_i = 0. \end{cases}$ In hardware, each pass-thru module consists of m two-input AND gates.

In Figure 1, the third module (*i.e.*, the rectangular shape α module) multiplies its input, which is an element of $GF(2^m)$, by α and reduces the result modulo $F(\alpha)$. Thus, this module is to essentially realize equation (7) in hardware. Since α is a root of $F(z)$,

$$F(\alpha) = \alpha^m + \sum_{i=0}^{m-1} f_i \alpha^i = 0. \tag{8}$$

Then multiplication of an arbitrary element $A \in GF(2^m)$ by α gives

$$A \cdot \alpha = (\sum_{i=0}^{m-1} a_i \alpha^i) \cdot \alpha = \sum_{i=0}^{m-1} a_i \alpha^{i+1} = \sum_{i=1}^{m-1} a_{i-1} \alpha^i + a_{m-1}\alpha^m. \tag{9}$$

Using (8) and (9), one can write

$$\begin{aligned} X &\triangleq A \cdot \alpha \bmod F(\alpha) \\ &= a_{m-1} \cdot f_0 + \textstyle\sum_{i=1}^{m-1} (a_{i-1} + a_{m-1} \cdot f_i)\alpha^i, \end{aligned} \tag{10}$$

where x_is are in $GF(2)$ and are the coordinates of X w.r.t. the PB. For any irreducible polynomial over $GF(2)$, $f_0 = 1$. Thus from (10), we write the coordinates of X as

$$x_i = \begin{cases} a_{i-1} + a_{m-1} \cdot f_i & 1 \le i \le m-1, \\ a_{m-1} & i = 0. \end{cases} \tag{11}$$

If ω is the Hamming weight of the irreducible polynomial $F(z)$, then the realization of (11) requires $\omega - 2$ XOR gates, and so does an α module. Thus, unlike the sum and pass-thru modules, the α module has a space (or circuit) complexity which depends on $F(z)$. The space complexity is minimum when $F(z)$ is a trinomial and maximum when $F(z)$ is an all-one-polynomial (AOP).

2.2 Fault Detection Strategy

In the following sections, we investigate fault detection schemes for $GF(2^m)$ multiplication operation. Towards this effort, the parity prediction method is used. This method is shown in Figure 2 where the CUT (circuit under test) block can be either a complete finite field multiplier or a part of it with A and B as inputs and Y as output, where $A, B \in GF(2^m)$. In this figure, the

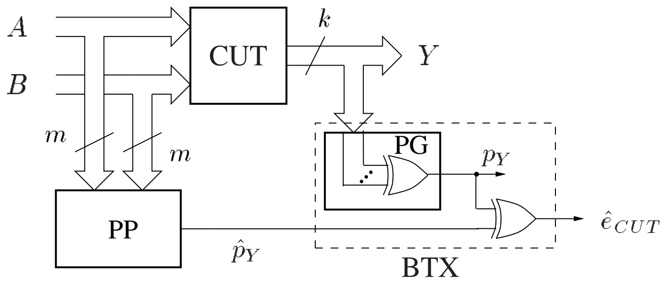


Figure 2: Error indication of the circuit under test (CUT) using parity prediction method.

parity generation (PG) block produces the actual parity of Y, i.e., $p_Y = \sum_{i=0}^{k-1} y_i$, where y_is are the coordinates of Y. The actual parity p_Y is then compared with the predicted parity $\hat{p}_Y$ using a single XOR gate as shown in the figure. This comparison is monitored by an error indicator flag $\hat{e}_{\text{CUT}}$, where $\hat{e}_{\text{CUT}} = 0$ indicates that no error has been detected and $\hat{e}_{\text{CUT}} = 1$ flags the detection of errors. The parity prediction (PP) block predicts the parity of the output Y using a function which depends only on the inputs of the CUT as $\hat{p}_Y = \Gamma_{\text{CUT}}(A, B)$. This function is hereafter referred to as *parity prediction* function. We assume that the parity of A and B (i.e., p_A and p_B, respectively) are available or they can be reliably pre-computed while loading the coordinates of A and B into the multiplier. We also assume that the PP and PG blocks can be made fault free or any fault in them can be detected using a suitable mechanism since these blocks are simple and/or regular. It is noted that the PG block and the last XOR gate in Figure 2 are implemented by a binary tree of XOR (BTX) gates. In the following sections, we derive the function Γ_{CUT} for each of the modules of the PB multipliers.

For the purpose of this investigation, we first consider $GF(2^m)$ multiplier circuit with a single fault. The single fault case provides simplicity in our analysis [7]. In the later part of the paper, we show that various types of multiple faults in the multiplier can also be detected. The following are basic assumptions for the fault model that we use in the entire paper: (i) a fault in a logical gate (i.e., XOR, AND, etc.) results in one of its inputs or the output being fixed to either a logic 0 (stuck-at-0, or s-a-0 in short) or a logic 1 (stuck-at-1, or s-a-1), respectively, and (ii) a fault can be either permanent or transient. In the following sections, we propose structures which detect the above mentioned faults during the operation in the field.

3 Fault Detections in Traditional Bit-Parallel Multiplier

3.1 Parity Predictions of Individual Modules

In the following, we obtain the parity prediction functions of the modules of the bit-parallel multiplier of Figure 1. PB multipliers that are capable of detecting faults are considered in the next sections.

3.1.1 Parity Prediction in α Module

Let ω be the Hamming weight of the irreducible polynomial $F(z)$. Then, (1) can be written as

$$F(z) = 1 + \sum_{j=1}^{\omega-2} z^{\rho_j} + z^m, \tag{12}$$

where ρ_j's are powers of z in (1) with $f_{\rho_j} = 1$, $1 \leq j \leq \omega - 2$. Then we have $1 \leq \rho_1 < \rho_2 < \rho_3 \cdots < \rho_{\omega-2} \leq m - 1$ and (11) can be written as

$$x_i = \begin{cases} a_{\rho_j - 1} + a_{m-1} & i = \rho_j,\ 1 \leq j \leq \omega - 2, \\ a_{i-1 \bmod m} & \text{otherwise.} \end{cases} \tag{13}$$

Using (13), a circuit diagram for the α module is shown in Figure 3(a). Note that a stuck-at fault in one of the $(\omega - 2)$ XOR gates of this module causes at most one error at the output.

For $j \in \{\rho_1, \rho_2, \cdots, \rho_{\omega-2}\}$, assume that the j-th gate in Figure 3(a) is faulty. Then all the output coordinates, except x_j, are error free. If the upper input of the j-th gate is stuck, then the erroneous j-th coordinate is

$$\dot{x}_j = \begin{cases} a_{m-1} & \text{for s-a-0,} \\ \overline{a}_{m-1} & \text{for s-a-1,} \end{cases} \tag{14}$$

where $\overline{x}$ indicates complement of x. On the other hand, if the lower input is stuck, then

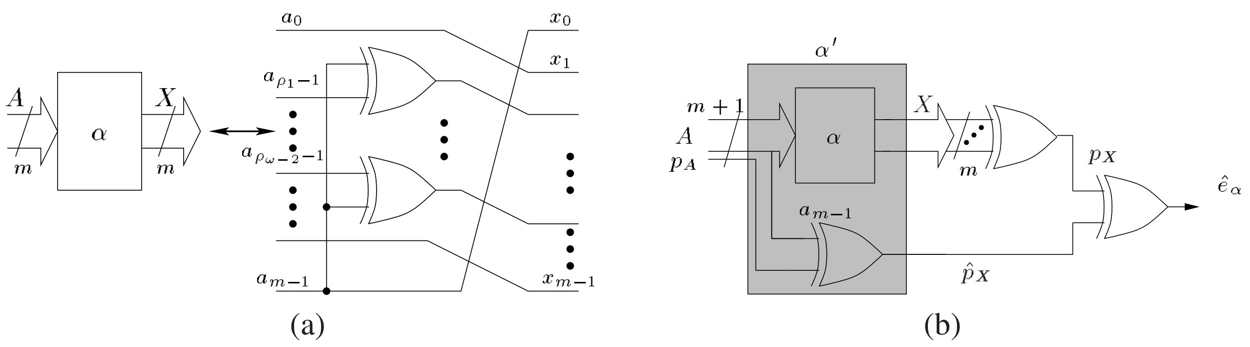


Figure 3: (a)The original circuit of α module. (b) The circuit for detecting a single fault (α' module).

$$\dot{x}_j = \begin{cases} a_{j-1} & \text{for} \quad \text{s-a-0,} \\ \overline{a}_{j-1} & \text{for} \quad \text{s-a-1.} \end{cases} \tag{15}$$

Detection of such faults are discussed below.

Assume that A and X are the input and output of the α module, respectively. Then we have the following lemma.

Lemma 1. *Let $X = A \cdot \alpha \bmod F(\alpha)$. Also, assume that $p_A = \sum_{i=0}^{m-1} a_i$ is the parity bit of A. Then, the predicted parity bit of X is*

$$\hat{p}_X = \Gamma_\alpha = p_A + a_{m-1} \in GF(2), \tag{16}$$

where a_{m-1} is the $(m-1)$-th coordinate of input A.

Proof. Using (11), $\hat{p}_X$ can be written as $\hat{p}_X = a_{m-1} + \sum_{i=1}^{m-1}(a_{i-1} + a_{m-1}f_i) = a_{m-1}\sum_{i=1}^{m-1} f_i + \sum_{i=0}^{m-1} a_i$.

Since $F(z)$ is irreducible over $GF(2)$, $F(z)$ is not divisible by $z+1$, and $F(1) = 1$. Then from (1), one obtains $\sum_{i=1}^{m-1} f_i = f_0 = 1$ and the proof is complete. □

It is noted that the authors of [2] have independently proved Lemma 1 and used it to develop a fault detection structure of AES.

Similar to (16), we can obtain the relation between parity bits of X and A in the fault free α

module as

$$p_X + p_A = a_{m-1}. \tag{17}$$

In the α module, a stuck-at fault in one of its gates will result in an output which is different from X and (16) will not hold. Thus, equation (16) can be used for detecting an error in the output of the α module. Circuit for detecting such errors is shown in Figure 3(b), where $\hat{e}_\alpha = 0$ indicates that no error has been detected and $\hat{e}_\alpha = 1$ flags the detection of an error. Since (16) is over $GF(2)$, the values of $\hat{e}_\alpha$ would detect not only a single error, but also any odd number of errors. With a similar argument, it is clear that an even number of errors are not detected by $\hat{e}_\alpha$.

3.1.2 Parity Predictions of Sum and Pass-thru Modules

The sum module of Figure 1 is a finite field adder which produces sum of the two elements of $GF(2^m)$ at its output. Let $A = (a_{m-1}, \cdots, a_1, a_0)$ and $B = (b_{m-1}, \cdots, b_1, b_0)$ be two inputs to this module. Then the output is $D = A + B = (d_{m-1}, \cdots, d_1, d_0)$ where $d_i = a_i + b_i$ for $0 \le i \le m-1$. The architecture of this module uses m two-input XOR gates. Let $p_A = \sum_{i=0}^{m-1} a_i$ and $p_B = \sum_{i=0}^{m-1} b_i$ be the parity bits of A and B, respectively. Then the parity bit of the output, $p_D = \sum_{i=0}^{m-1} d_i$, is predicted by

$$\hat{p}_D = \Gamma_{sum} = p_A + p_B \tag{18}$$

using one extra XOR gate. Let us denote these $m+1$ XOR gates as the new sum module.

The pass-thru module of Figure 1 multiplies an element $A \in GF(2^m)$ by a single bit $b \in GF(2)$ which can be implemented using m two-input AND gates. Let $G \in GF(2^m)$ be the output of such a module with inputs of A and b. Thus, the output of this module G is zero when $b = 0$ and A when $b = 1$.

Detection of an odd number of errors is accomplished by using a single parity bit similar to the α and sum modules. Let A and $p_A = \sum_{i=0}^{m-1} a_i$ be the input of the pass-thru module and its parity bit respectively. Then, the parity bit of the output $G = bA$ is $p_G = \sum_{i=0}^{m-1} g_i$, where $g_i = b \cdot a_i$, $0 \le i \le m-1$, are the coordinates of G. Thus, the predicted parity bit of the output can be expressed as

$$\hat{p}_G = \Gamma_{pass} = b \cdot p_A \tag{19}$$

which requires only one AND gate for its implementation. Let us denote the original pass-thru module together with this AND gate as the new pass-thru module similar to the new sum module. These new modules are used in the next section.

The discussions in this section have so far dealt with the parity prediction functions of individual modules. Using these parity functions, below we attempt to detect faults in the entire multiplier.

3.2 Fault Detection Architectures

When α modules are cascaded as shown in the array of α modules in Figure 1, we can re-state Lemma 1 as follows:

Lemma 1a. *Using the symbols defined earlier, we have*

$$\hat{p}_{X^{(j)}} = p_A + \sum_{k=0}^{j-1} x_{m-1}^{(k)}, \qquad j = 1, 2, \cdots, m-1, \tag{20}$$

where $x_{m-1}^{(k)}$ is the $(m-1)$-th coordinate of $X^{(j)} = A\alpha^j \mod F(\alpha)$ w.r.t. the polynomial basis, respectively.

Proof. The proof can be done by induction as follows. Equation (20) is the same as (16) when $j = 1$. Let us assume equation (20) holds for $j = l, 1 < l < m-1$, i.e., $\hat{p}_{X^{(l)}} = p_A + \sum_{k=0}^{l-1} x_{m-1}^{(k)}$, then we will prove that (20) holds for $j = l+1$. This can be done by using (16) for the $l-$th α module as

$$\hat{p}_{X^{(l+1)}} = p_{X^{(l)}} + x_{m-1}^{(l)}. \tag{21}$$

Now, if we assume that no fault occurs in the α modules, then $\hat{p}_{X^{(l)}} = p_{X^{(l)}} = p_A + \sum_{k=0}^{l-1} x_{m-1}^{(k)}$, and by substituting in (21), the proof is complete. $\square$

Then, the parity bit of the output of the polynomial basis multiplier can be predicted using the following theorem.

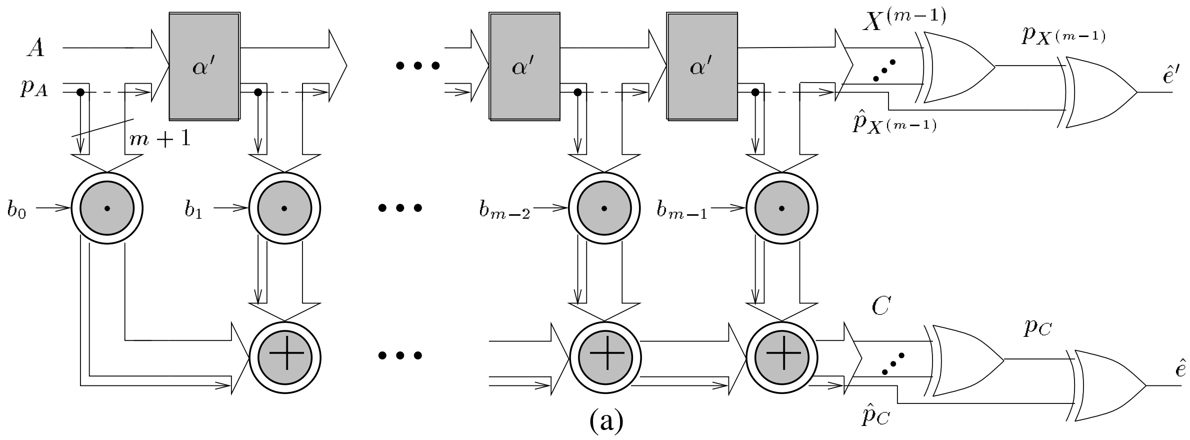
Theorem 1. *Let C be the product of two arbitrary elements A and B of $GF(2^m)$. Then,*

$$\hat{p}_C = \sum_{j=0}^{m-1} b_j \hat{p}_{X^{(j)}}. \tag{22}$$

Proof. Using (6) and generalizing (18) from two inputs to m inputs, one can obtain $\hat{p}_C = \sum_{j=0}^{m-1} \hat{p}_{b_j X^{(j)}}$. Now, using (19), one can see that $\hat{p}_{b_j X^{(j)}} = b_j \hat{p}_{X^{(j)}}$, and the proof is complete. $\square$

In order to detect a single fault in a finite field multiplier, equation (22) can be realized easily by replacing all the α, pass-thru and sum modules in Figure 1 with the α' module and the new pass-thru and sum modules as shown in Figure 4(a) (these three new modules are shaded in this figure to distinguish them from the old ones). The bus width of this multiplier is $m+1$. For the purpose of illustration, the fault detection architecture for $GF(2^3)$ multiplier generated by irreducible polynomial $F(z) = z^3 + z + 1$ is shown in Figure 4(b). Since the output of any gate of the shaded pass-thru and sum modules in Figure 4(a) is connected to only one gate, the single stuck fault at any gate of these modules changes only one coordinate of the output of this multiplier. Therefore, a circuit that compares the actual parity p_C with the predicted $\hat{p}_C$ to generate $\hat{e}$, which is shown at the bottom end of the figure, is capable of detecting any single fault in the shaded sum and pass-thru modules of Figure 4(a). Also, it is clear that any single fault in any XOR gate in the parity generation circuit p_C and the very last XOR gate can be detected by $\hat{e}$. It is noted that such a comparison circuit is implemented using a binary tree of XOR gates which is done by three last XOR gates at the bottom end of Figure 4(b).

It is worth mentioning that this circuit can also detect certain multiple faults that result in an odd number of errors at the output. To give an example of such multiple faults, let us re-arrange all shaded sum and pass-thru modules of Figure 4(a) into $m+1$ blocks of inner product cells, each of which contains m AND gates and $m-1$ XOR gates and generates one output of the multiplier c_i, $0 \leq i \leq m-1$, as well as the parity prediction. Then, any multiple faults in an inner product cell



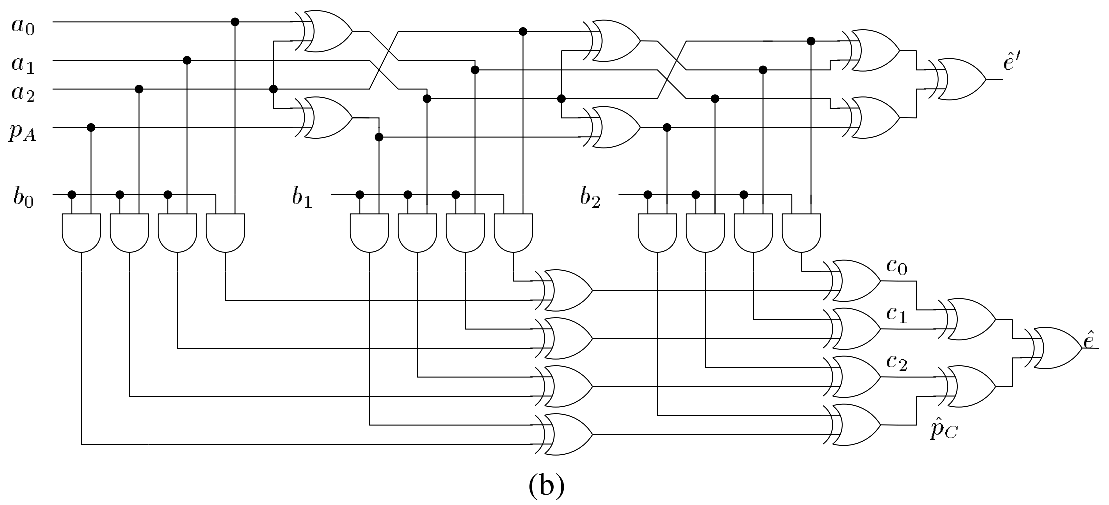


Figure 4: (a) Multiplication of two elements in $GF(2^m)$ with fault detection capability. (b) An example for $GF(2^3)$ generated by $z^3 + z + 1$.

may change the corresponding bit and it will be detected by $\hat{e}$. It is noted that Figure 4 can detect transient faults including single and the above mentioned multiple stuck-at faults.

One important drawback of the multiplier shown in Figure 4(a) is that it can detect a single stuck-at-fault in the right-most α' module, but may not in other α' modules of Figure 4(a). This is because a stuck-at fault in any α' modules other the right-most one is most likely to change more than one bit of the multiplier output. Then, these errors cannot be detected if an even number of output bits are changed due to a single fault in the α' array.

In this connection, note that the α' modules require $(\omega - 1)(m - 1)$ XOR gates. For the cryptographic applications, the most commonly used irreducible polynomials are trinomial $(\omega = 3)$ and pentanomial ($\omega = 5$). Also, one can easily determine that the shaded pass-thru and sum modules consist of $m(m+1)$ AND gates and $(m-1)(m+1) = m^2 - 1$ XOR gates, respectively. Therefore, for such applications where the value of m is large, the space complexity of the α' modules is negligible compared with space complexity of the whole multiplier and then it is more likely that a single fault is not located in a α' module, but in a pass-thru or sum module. In such a case, the error due to the fault can be detected as explained earlier.

Nevertheless, in order to detect a single fault in the α' modules, one can use another flag, i.e.,

$$\hat{e}' = \hat{p}_{X^{(m-1)}} + p_{X^{(m-1)}}\,, \tag{23}$$

where $\hat{p}_{X^{(m-1)}}$ is obtained by using (20) for $j = m-1$ as

$$\hat{p}_{X^{(m-1)}} = \Gamma_{\alpha\, array} = p_A + \sum_{k=0}^{m-2} x_{m-1}^{(k)}. \tag{24}$$

Equation (23) is realized at the top right end of Figure 4(a) and then $\hat{e}'$ can be implemented by a binary tree of XOR gates as shown in Figure 4(b). Below, we explain how it flags a single fault in an α' module.

As shown in Figure 3(b), each α' module consists of the original α module and an XOR gate. We assume that a single fault changes only one bit of the output of the α' module. Then, this fault will be detected by $\hat{e}'$ in Figure 4, because either a predicted parity or the output of an α module will be changed as a result of such a single fault. If a predicted parity is changed, then $\hat{p}_{X^{(m-1)}}$ will be changed and it will be indicated by $\hat{e}' = 1$. Otherwise, $p_{X^{(m-1)}}$ is changed which results in $\hat{e}' = 1$. This is because of the following.

Equation (17) can be generalized for detecting an error in each of the α modules as

$$p_{X^{(j+1)}} + p_{X^{(j)}} = x_{m-1}^{(j)}, \qquad j = 0, 1, \cdots, m-2\,, \tag{25}$$

where $X^{(0)} = A$. For $1 \le i \le m-1$, suppose the fault is in the α module located in the i-th α' module of Figure 4. Then the outputs of the α modules are correct for $1 \le k \le i-1$ and may be in error for $i \leq k \leq m-1$. Thus, we can divide (25) into the following three parts.

$$\begin{aligned} p_{X^{(k)}} + p_{X^{(k-1)}} &= x_{m-1}^{(k)}, \qquad 0 \leq k \leq i-1, & (26) \\ p_{X^{(i)}} + p_{X^{(i-1)}} &= x_{m-1}^{(i-1)} + \hat{e}', & (27) \\ p_{X^{(k)}} + p_{X^{(k-1)}} &= x_{m-1}^{(k)}, \qquad i+1 \leq k \leq m-1. & (28) \end{aligned}$$

Equation (26) is similar to (25), because there is no fault up to the $(i-1)$-th α module. Since the fault is in the i-th module, (25) is not valid and this is shown by adding a single bit, $\hat{e}'$, to the right hand side of (27). This flag is 1 when this fault produces an error at the output of this module. After the i-th α module, although not all inputs and outputs are correct but (28) is still valid, since according to our assumption these modules are fault free. By adding these three equations for $0 \leq k \leq m-1$ and using $X^{(0)} = A$, the relation of (23) is obtained. Note that $\hat{e}'$ in (23) is independent of the position of the faulty module.

3.2.1 Fault Detection Using Double Parity Bits

The structure shown in Figure 4(a) does not make use of the parity bit of B, i.e., p_B is not used as an input of the multiplier. Thus, this structure can be used even when the parity of only one of the two inputs is available. If the parity bits of both inputs are available, then the parity bit $\hat{p}_C$ of the product can be obtained in an alternate way as presented below.

Corollary 1. *Let p_A and p_B be the parity bits of A and B, respectively. Then predicted parity bit of $C = AB$ is*

$$\hat{p}_C = p_A p_B + \sum_{j=1}^{m-1} b_j \sum_{k=0}^{j-1} x_{m-1}^{(k)}. \qquad (29)$$

Proof. Substituting (20) into (22), we have

$$\begin{aligned} \hat{p}_C &= b_0 p_A + \sum_{j=1}^{m-1} b_j \left(p_A + \sum_{k=0}^{j-1} x_{m-1}^{(k)}\right) & (30) \\ &= p_A \sum_{j=0}^{m-1} b_j + \sum_{j=1}^{m-1} b_j \sum_{k=0}^{j-1} x_{m-1}^{(k)} \end{aligned}$$

$$= \quad p_A p_B + \sum_{j=1}^{m-1} b_j \sum_{k=0}^{j-1} x_{m-1}^{(k)}.$$

$\square$

As seen in (29), the only dependency of $\hat{p}_C$ on b_0 is through p_B. Also, note that (29) is similar to (30) and then by modifying Figure 4(a) (and Figure 4(b) for $GF(2^3)$), one can obtain another structure. This structure, which is denoted as double parity fault detection structure, is shown in Figure 5(a). This new structure is obtained by replacing two shaded modules, i.e., the first pass-thru and α' modules, of Figure 4(a) with their corresponding non-shaded modules. Also, the output of the AND gate which generates $p_A p_B$ is connected to the input of next shaded sum module. This bit is located at the last position and predicts the output parity. Since p_A is removed from the input of the α module, it should be XORed at the output to obtain $\hat{p}_{X^{(m-1)}}$ as shown in Figure 5(a). The corresponding architecture for $GF(2^3)$, which can also be obtained from Figure 4(b), is shown in Figure 5(b). It is noted that Theorem 1 and Corollary 1 are not restricted to any particular irreducible polynomials.

It is interesting to compare Corollary 1 with the parity prediction of a conventional integer multiplier that multiplies two m-bit integers, say A and B, to generate a $2m$-bit output, say S. Then, the predicted parity can be obtained as $\hat{p}_S = p_A p_B + p_C$, where p_C is the carry parity and can be obtained by adding all the carry bits [11].

4 Fault Detection in Low Complexity Bit-Parallel Multiplier

Recently, a new class of low complexity multiplier has been proposed in [17] (for trinomials), [15] (for special pentanomials) and [14] (for general irreducible polynomials). The multiplier structure proposed in [14] is shown in Figure 6(a). As seen in the figure, it consists of two blocks, namely I and Q. After substituting (2) into (4), one can see that S is a binary polynomial of degree $2m-2$

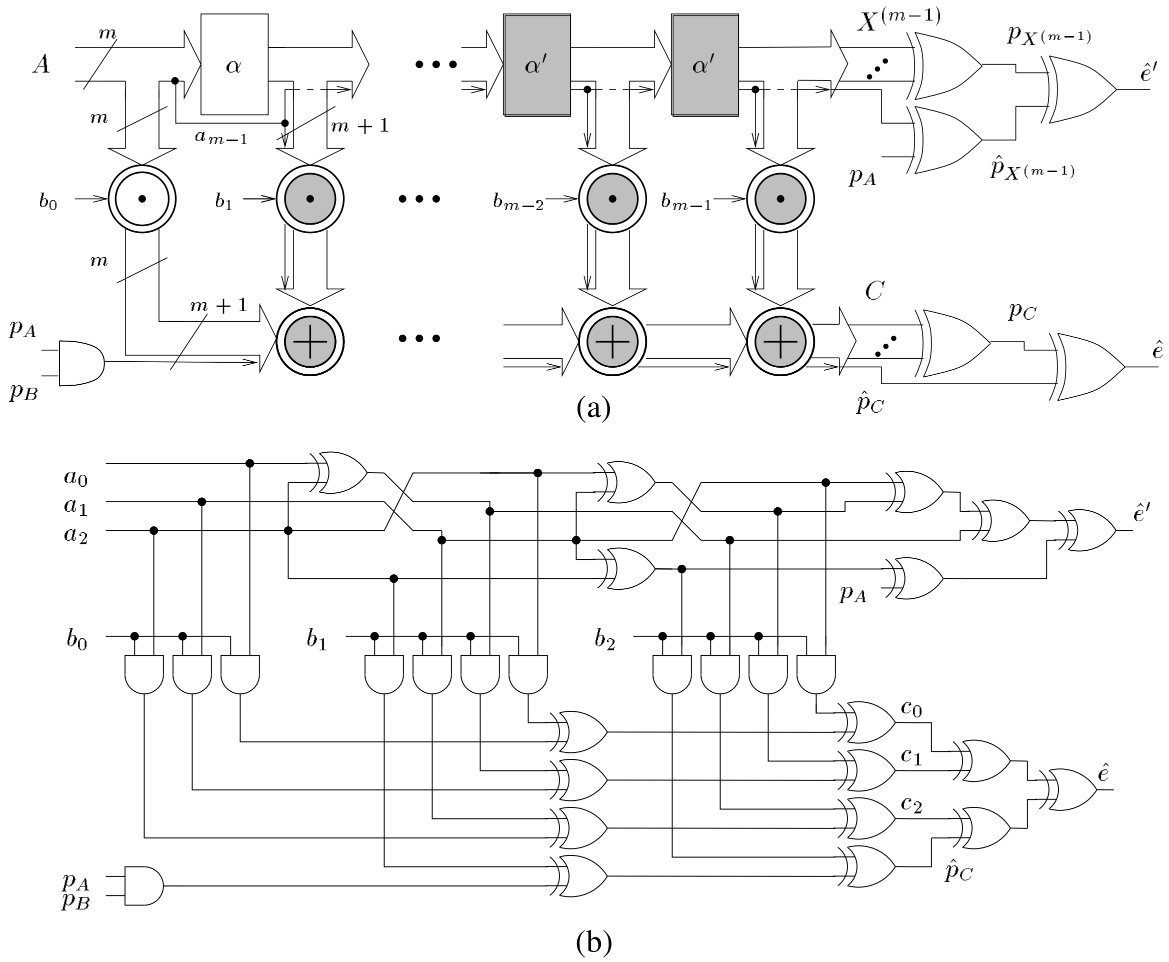


Figure 5: (a) Fault detection structure using double parity bits. (b) An example for $GF(2^3)$.

as

$$S = \sum_{i=0}^{m-1} d_i \alpha^i + \sum_{j=0}^{m-2} e_j \alpha^{m+j} \tag{31}$$

and the first block I of Figure 6(a) generates all coefficients of S using

$$d_i = [a_i,\, a_{i-1},\, \cdots,\, a_0,\, 0, \cdots, 0][b_0,\, b_1,\, \cdots,\, b_{m-1}]^T, \;\; 0 \le i \le m-1, \tag{32}$$

and

$$e_j = [0, \cdots, 0,\, a_{m-1},\, a_{m-2},\, \cdots,\, a_{j+1},][b_0,\, b_1,\, \cdots,\, b_{m-1}]^T, \;\; 0 \le j \le m-2, \tag{33}$$

which requires m^2 AND gates and $(m-1)^2$ XOR gates. Then, the second block Q in Figure 6(a) reduces these $2m-1$ coordinates to the m coordinates of C using the following matrix equation [14]

$$\mathbf{c} = [c_0,\, c_1,\, \cdots,\, c_{m-1}]^T = \mathbf{d} + \mathbf{Q}^T \mathbf{e}, \tag{34}$$

where $\mathbf{d} = [d_0,\, d_1,\, \cdots,\, d_{m-1}]^T$ and $\mathbf{e} = [e_0,\, e_1,\, \cdots,\, e_{m-2}]^T$. In (34), $\mathbf{Q}$ is an $m-1$ by m binary matrix, known as the *reduction matrix,* which is obtained from [9].

$$[\alpha^m,\, \alpha^{m+1},\, \cdots,\, \alpha^{2m-2}]^T \equiv \mathbf{Q}[1,\, \alpha,\, \cdots,\, \alpha^{m-1}]^T \pmod{F(\alpha)}. \tag{35}$$

As seen in (35), the reduction matrix is fixed for a given irreducible polynomial $F(\alpha)$. For the example of $F(z) = z^3 + z + 1$, one can obtain $\mathbf{Q} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$. Thus, the corresponding multiplier over $GF(2^3)$ is shown in Figure 6(b).

In this section, we want to detect a fault in the low complexity multiplier architecture of Figure 6. One way to achieve this is to add circuits at the output of the multiplier so that parity bit $\hat{p}_C$ is generated based on equation (22) or (29). The following will help us to implement either (22) or (29).

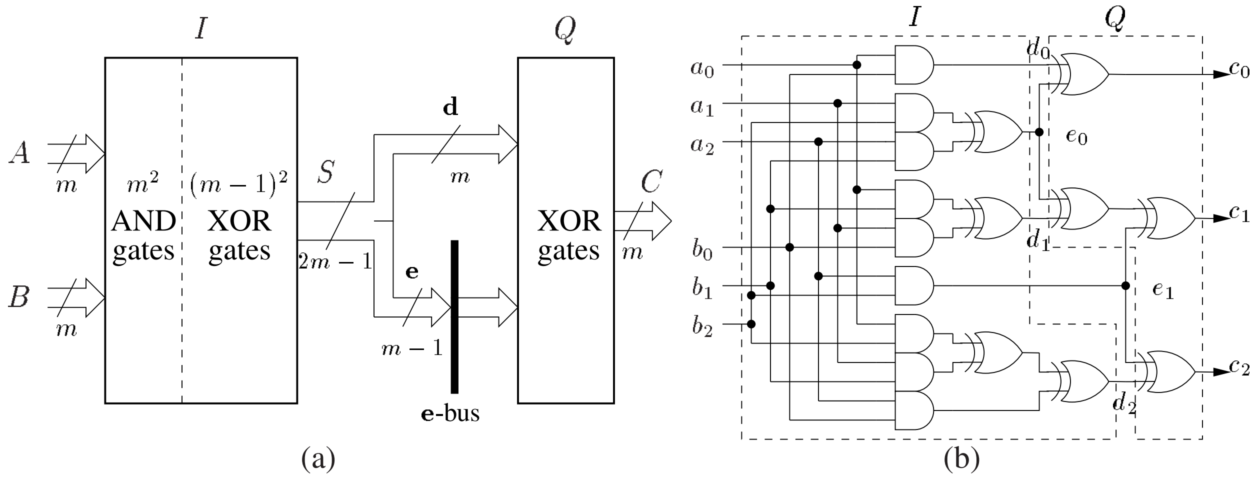


Figure 6: Low complexity bit-parallel multiplier: (a) general structure over $GF(2^m)$, (b) an example over $GF(2^3)$.

Lemma 2. *Let* $\mathbf{q}_{m-1}$ *be the right-most column of the reduction matrix* $\mathbf{Q}$*. Then,*

$$x_{m-1}^{(k)} = a_{m-k-1} + [a_{m-k},\ a_{m-k+1}, \cdots,\ a_{m-1},\ \underbrace{0, 0, \cdots, 0}_{m-1-k}\ ]\mathbf{q}_{m-1}. \tag{36}$$

Proof. Using the multiplication formulation of (34), one can obtain the coordinates of $X^{(k)} = A\alpha^k$ as

$$\begin{aligned}&[x_0^{(k)},\ x_1^{(k)}, \cdots,\ x_{m-1}^{(k)}] = \mathbf{d}^T + \mathbf{e}^T\mathbf{Q}\\ &= [\ \underbrace{0, 0, \cdots, 0,}_{k}\quad a_0,\ a_1, \cdots,\ a_{m-k-1}] + [a_{m-k},\ a_{m-k+1}, \cdots,\ a_{m-1},\ \underbrace{0, 0, \cdots, 0}_{m-1-k}\ ]\mathbf{Q}\end{aligned}$$

and its last coordinate is $x_{m-1}^{(k)}$ as given in (36). □

By implementing either (22) or (29) and using (36), any single stuck faults in the path to generate $\mathbf{d}$ will change at most one bit in the output of C and will be detected (see Figure 6(b)). However, it cannot detect single faults in the I block that generates one of the entries of $\mathbf{e}$. This is because such a fault may cause even number of errors at the output. For example, as seen in Figure 6(b), an error in e_0 (or e_1) causes two errors in c_0 and c_1 (or c_1 and c_2). Moreover, this fault detection structure is not area efficient because all $x_{m-1}^{(k)}$s should be implemented. In the following, we will present an efficient fault detection structure for Figure 6(a).

4.1 Formulation

In order to obtain a fault detection structure for the multiplier of Figure 6(a), we use two parity prediction functions, namely Γ_I and Γ_Q, to detect single faults in the I and Q blocks, respectively. These functions can be obtained from the following lemmas.

Lemma 3. *Let A and B be two field elements and S be their multiplication (without modular reduction). Then,*

$$\hat{p}_S = \Gamma_I(A, B) = p_A p_B, \tag{37}$$

where p_A and p_B are the parity bits of A and B, respectively.

Proof. As seen in (31), S is a binary polynomial of degree $2m-2$. Then, one can see that $\hat{p}_S = p_{\mathbf{d}} + p_{\mathbf{e}}$, where

$$p_{\mathbf{d}} = \sum_{i=0}^{m-1} d_i, \quad p_{\mathbf{e}} = \sum_{i=0}^{m-2} e_i. \tag{38}$$

Using (32) and (33) (assume $e_{m-1} = 0$), one can complete the proof as

$$\hat{p}_S = \sum_{i=0}^{m-1}(d_i+e_i) = \sum_{i=0}^{m-1}[a_i, \cdots, a_0, , a_{m-1} \cdots, a_{i+1}]\mathbf{b} = [p_A, p_A, \cdots, p_A]\mathbf{b} = p_A \sum_{i=0}^{m-1} b_i = p_A p_B.$$

□

Lemma 4. *Let $\mathbf{q}_i$, $0 \le i \le m-1$, be the column i of the reduction matrix $\mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, \cdots, \mathbf{q}_{m-1}]$. Then,*

$$\hat{p}_C = \Gamma_Q(\mathbf{d}, \mathbf{e}) = p_{\mathbf{d}} + \mathbf{e}^T\mathbf{q}, \tag{39}$$

where $p_{\mathbf{d}}$ is given in (38) and $\mathbf{q} = \sum_{i=0}^{m-1} \mathbf{q}_i$.

Proof. Using (34) and the linear property of the parity function, we have $\hat{p}_C = p_{\mathbf{d}} + p_{\mathbf{Q}^T\mathbf{e}}$ and the proof is complete since $p_{\mathbf{Q}^T\mathbf{e}} = p_{\mathbf{e}^T\mathbf{Q}} = \sum_{i=0}^{m-1} \mathbf{e}^T\mathbf{q}_i = \mathbf{e}^T \sum_{i=0}^{m-1} \mathbf{q}_i = \mathbf{e}^T\mathbf{q}$. □

Corollary 2. *Let* $\mathbf{q}' = \mathbf{q} + \mathbf{1}_{m-1\times 1}$, *where* $\mathbf{1}_{m-1\times 1}$ *is a column vector whose* $m-1$ *entries are all one. Then,*

$$\hat{p}_C = \Gamma_Q(\mathbf{d}, \mathbf{e}) = p_A p_B + + \mathbf{e}^T \mathbf{q}'. \tag{40}$$

Proof. Using (39) and $p_{\mathbf{e}} = \sum_{i=0}^{m-2} e_i = \mathbf{e}^T \mathbf{1}_{m-1\times 1}$, we can write $\hat{p}_C = p_{\mathbf{d}} + p_{\mathbf{e}} + \mathbf{e}^T \mathbf{1}_{m-1\times 1} + \mathbf{e}^T \mathbf{q}$ and this completes the proof since $p_{\mathbf{d}} + p_{\mathbf{e}} = p_A p_B$. □

4.2 Fault Detection Architecture

Using Lemmas 3 and 4 and comparing the predicted parity bits of $\hat{p}_S$ and $\hat{p}_C$ with the actual parity bits p_S and p_C, respectively, we will be able to detect any single faults (permanent or transient) occurred in the I and Q blocks of Figure 6(a). The corresponding fault detection structure is shown in Figure 7(a). In this figure two flags $\hat{e}_I = p_S + \hat{p}_S$ and $\hat{e}_Q = p_C + \hat{p}_C$ indicate the presence of a single fault in the I and Q blocks of Figure 6(a), respectively. In the fault free situation both flags are zero. Also, the overhead needed for $GF(2^3)$ multiplier of Figure 6(b) to make it fault detectable is shown in Figure 7(b).

Assume that there is a single stuck-at-fault in the I block of Figure 6(a), then at most one error will be in S (either e_i, $0 \le i \le m-2$, or d_j, $0 \le j \le m-1$ will be changed). This is because the output of any gate (either an AND gate or an XOR gate) in the I block is connected only to an input of the next gate (which is an XOR gate). Thus, if this fault changes the input bit of the next gate, then only one of the bits of S (see Figure 7(b)) will be changed. As a result, the actual parity of S, i.e.,

$$p_S = \sum_{j=0}^{m-1} d_j + \sum_{i=0}^{m-2} e_i = p_{\mathbf{d}} + p_{\mathbf{e}} = p_{\mathbf{d}} + \mathbf{e}^T \mathbf{1}_{m-1\times 1} = p_{\mathbf{d}} + \mathbf{e}^T \mathbf{q} + \mathbf{e}^T \mathbf{q}', \tag{41}$$

will be changed. In Figure 7(a), equation (41) is realized and is compared with the predicted parity of (37), i.e., $\hat{p}_S = p_A p_B$, to indicate at most one error due to a fault in the I block as $\hat{e}_I = p_S + \hat{p}_S = 1$.

Now, we assume that a single stuck-at-fault (either permanent or transient) occurs in the Q

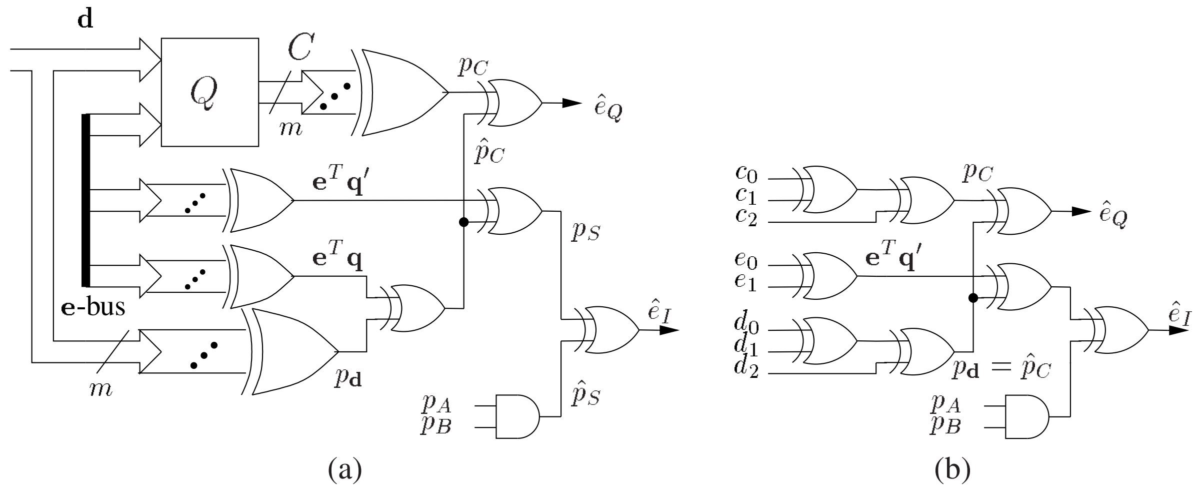


Figure 7: Fault detection of the low complexity multiplier in Figure 6, (a) over $GF(2^m)$, (b) over $GF(2^3)$ generated by $z^3 + z + 1$.

block of Figure 6(a), then all e_is and d_js are error free. This implies that in Figure 7(a), $\hat{e}_I = 0$ and $\hat{p}_C = p_\mathbf{d} + \mathbf{e}^T\mathbf{q}$ (see (39)) is error free. It is noted that although some lines on the e-bus are reused (both e_0 and e_1 in Figure 7(b) are re-used), no signals inside the Q block of the multiplier are re-used. Thus, the output of each XOR gate of the Q block is connected to a single XOR gate. As a result, if a fault (either permanent or transient) occurs in this block, it will cause at most one bit error in C. This will change p_C in Figure 7(a) and it will be flagged by $\hat{e}_Q$, because $\hat{p}_C$ is error free.

It is worth mentioning that the above fault detection structure can also detect multiple faults in any number of gates that generate d_i (or e_i). This is because they may change at most one bit of d_i (or e_i) and it will be detected by $\hat{e}_I$.

4.3 Special Irreducible Polynomials

In (39) and (40) the entries of $\mathbf{q}$ and $\mathbf{q}'$ belong to GF(2) and are determined based on the underlying irreducible polynomials. Below, we want to obtain $\mathbf{q}$ and hence $\hat{p}_C = \Gamma_Q(\mathbf{d}, \mathbf{e})$ for three classes of irreducible polynomials.

Equally-Spaced Polynomials

For an s equally-spaced polynomial of degree m, i.e., $F(z) = z^{ns} + z^{(n-1)s} + \cdots + z^s + 1$, $ns = m$, the reduction matrix $\mathbf{Q}$ is [14]

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_{s\times s}, \quad \mathbf{I}_{s\times s}, \quad \cdots, \quad \mathbf{I}_{s\times s} \\ \mathbf{I}_{m-s-1\times m-s-1}, \quad \mathbf{0}_{m-s-1\times s+1} \end{bmatrix}, \tag{42}$$

where $\mathbf{I}_{j\times j}$ is the $j \times j$ unity matrix and $\mathbf{0}_{m-s-1\times s+1}$ is a zero matrix which has $m-s-1$ rows and $s+1$ columns. Then, by adding columns of (42), one can obtain

$$\mathbf{q} = \begin{cases} \mathbf{1}_{m-1\times 1} & \text{if } n \text{ odd} \\ \begin{bmatrix} \mathbf{0}_{s\times 1} \\ \mathbf{1}_{m-s-1\times 1} \end{bmatrix} & \text{otherwise,} \end{cases}, \quad \mathbf{q}' = \begin{cases} \mathbf{0}_{m-1\times 1} & \text{if } n \text{ odd} \\ \begin{bmatrix} \mathbf{1}_{s\times 1} \\ \mathbf{0}_{m-s-1\times 1} \end{bmatrix} & \text{otherwise.} \end{cases} \tag{43}$$

Therefore, using $\mathbf{q}$ in (43) and (39), we can state the following.

Lemma 5. *For an s equally-spaced polynomial of degree m, i.e., $F(z) = z^{ns} + z^{(n-1)s} + \cdots + z^s + 1$, $ns = m$, the parity prediction function is $\hat{p}_C = \Gamma_Q(\mathbf{d}, \mathbf{e}) = p_{\mathbf{d}} + \sum_{i=t}^{m-2} e_i$, where $t = \begin{cases} 0, & \text{if } n \text{ odd} \\ s, & \text{otherwise.} \end{cases}$*

Remark 1. *If n is odd, one can see $t = 0$ and $\hat{p}_C = p_{\mathbf{d}} + \sum_{i=0}^{m-2} e_i = p_{\mathbf{d}} + p_{\mathbf{e}} = p_A p_B$.*

Remark 2. *For all-one polynomials, where $s = 1$ and $n = m$ is even, one can see $t = 1$ and $\hat{p}_C = p_A p_B + e_0$ (using (40) and $\mathbf{q}'$ in (43)). Since $e_0 = \sum_{j=1}^{m-1} b_j a_{m-j}$ (see (33)), one can simplify the parity prediction of the multiplication to $\hat{p}_C = p_A p_B + \sum_{j=1}^{m-1} b_j a_{m-j}$. This matches the corresponding result reported in [5].*

Trinomials

Let $F(z) = z^m + z^k + 1$ be an irreducible trinomial generating $GF(2^m)$. Irreducible trinomials have drawn significant attention in the past and reference [16] lists an irreducible trinomial for every degree m ($\leq 10,000$) for which such a polynomial exists. To obtain the predicted parity function of the multiplication, we now derive $\mathbf{Q}$ for the trinomial $F(z) = z^m + z^k + 1, 1 \leq k < \frac{m}{2}$. This class of trinomials is important because two trinomials recommended by NIST for ECDSA, namely $z^{233} + z^{74} + 1$ and $z^{409} + z^{87} + 1$, belong to this class.

Lemma 6. *For the trinomial $F(z) = z^m + z^k + 1$, $1 \leq k < \frac{m}{2}$, the parity prediction function is*

$$\hat{p}_C = \Gamma_Q(\mathbf{d}, \mathbf{e}) = \begin{cases} p_{\mathbf{d}} + \sum_{i=m-k}^{m-2} e_i & \textit{if } k > 1 \\ p_{\mathbf{d}} & \textit{if } k = 1. \end{cases}$$

Proof. Since $\alpha^{m+j} = \begin{cases} \alpha^j + \alpha^{j+k} & \text{if } 0 \leq j < m-k \\ \alpha^j + \alpha^{j-m+k} + \alpha^{j-m+2k} & \text{if } m-k \leq j < m-1, \end{cases}$ one can see that

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_{m-1 \times m-1}, & \mathbf{0}_{m-1 \times 1} \end{bmatrix} + \begin{bmatrix} & \mathbf{0}_{m-k \times k}, \quad \mathbf{I}_{m-k \times m-k} & \\ \mathbf{I}_{k-1 \times k-1}, \quad \mathbf{0}_{k-1 \times 1}, & \mathbf{I}_{k-1 \times k-1}, & \mathbf{0}_{k-1 \times m-2k-1} \end{bmatrix}$$

Now, using the representation of $\mathbf{Q}$, we can obtain $\mathbf{q} = \mathbf{1}_{m-1 \times 1} + \begin{bmatrix} \mathbf{1}_{m-k \times 1} \\ \mathbf{0}_{k-1 \times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{m-k \times 1} \\ \mathbf{1}_{k-1 \times 1} \end{bmatrix}$ which completes the proof if (39) is used. □

Pentanomials

After trinomials, pentanomials are the most suitable class of polynomials for defining the finite field $GF(2^m)$. A polynomial with five non-zero coefficients, i.e., $F(z) = z^m + z^{k_3} + z^{k_2} + z^{k_1} + 1$, where $1 \leq k_1 < k_2 < k_3 \leq m - 1$, is called a *pentanomial* of degree m. In terms of the values of k_1, k_2, and k_3, it can be divided to different classes of pentanomials. Here, we consider a popular class of pentanomial where $k_3 < \frac{m}{2}$. For example, among the five binary fields recommended by NIST for ECDSA, three are pentanomials that belong to this class, i.e., $z^{163} + z^7 + z^6 + z^3 + 1$, $z^{283} + z^{12} + z^7 + z^5 + 1$, and $z^{571} + z^{10} + z^5 + z^2 + 1$.

Lemma 7. *For the pentanomial $F(z) = z^m + z^{k_3} + z^{k_2} + z^{k_1} + 1$, where $1 \leq k_1 < k_2 < k_3 < \frac{m}{2}$, the* *parity prediction function is* $\hat{p}_C = \Gamma_Q(\mathbf{d},\mathbf{e}) = \begin{cases} p_{\mathbf{d}} + \sum_{i=m-k_1}^{m-2} e_i + \sum_{i=m-k_3}^{m-k_2-1} e_i & \text{if } k_1 > 1 \\ p_{\mathbf{d}} + \sum_{i=m-k_3}^{m-k_2-1} e_i & \text{if } k_1 = 1. \end{cases}$

Proof. In [14], the reduction matrix $\mathbf{Q}$ is obtained as $\mathbf{Q} = \mathbf{Q}_0 + \mathbf{Q}_1 + \mathbf{Q}_2 + \mathbf{Q}_3$ for pentanomials $P(x) = x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$, where $1 < k_1 < k_2 < k_3 \le \frac{m}{2}$. The graphical representations of these sub-matrices are shown in Figures 6(a)-(d) of [14]. Let $\mathbf{q}^{(01)}$ and $\mathbf{q}^{(23)}$ be two column vectors that can be obtained by adding the columns of $\mathbf{Q}_0 + \mathbf{Q}_1$ and $\mathbf{Q}_2 + \mathbf{Q}_3$, respectively. Then, using Figures 6(a) and 6(b) of [14], we can obtain $\mathbf{q}^{(01)} = \begin{bmatrix} \mathbf{0}_{m-k_1 \times 1} \\ \mathbf{1}_{k_1-1 \times 1} \end{bmatrix}$. Similarly, using Figures 6(c) and 6(d) of [14], one can obtain $\mathbf{q}^{(23)} = \begin{bmatrix} \mathbf{0}_{m-k_3 \times 1} \\ \mathbf{1}_{k_3-k_2 \times 1} \\ \mathbf{0}_{k_2-1 \times 1} \end{bmatrix}$. Since $\mathbf{q} = \mathbf{q}^{(01)} + \mathbf{q}^{(23)}$, then $\mathbf{e}^T\mathbf{q} = \mathbf{e}^T\mathbf{q}^{(01)} + \mathbf{e}^T\mathbf{q}^{(23)} = \sum_{i=m-k_1}^{m-2} e_i + \sum_{i=m-k_3}^{m-k_2-1} e_i$ and the proof is complete for $k_1 > 1$. Similar proof can be done for $k_1 = 1$. □

5 Fault Detection in Bit-Serial Multipliers

There are two types of bit-serial, namely LSB-first and MSB-first, multipliers. Below, we consider these multipliers and present their fault detection architectures.

5.1 LSB First Multiplier

The PB multiplier of Figure 1 can be realized in a bit-serial fashion as shown in Figure 8(a). In this multiplier structure, both $X = (x_{m-1}, \cdots, x_1, x_0)$ and $Y = (y_{m-1}, \cdots, y_1, y_0)$ are m bit registers. Let $X(n)$ and $Y(n)$ denote the contents of X and Y at the n-th, $0 \le n \le m$, clock cycle, respectively. Suppose the X register is initialized with A, i.e., $X(0) = A$, then the content of this register at the n-th clock cycle is $X(n) = X^{(n)}$, where $X^{(n)} \in GF(2^m)$ is defined in (7). Also, suppose that the register Y is initially cleared, i.e., $Y(0) = 0$. Then, one can obtain the content of Y at the first clock cycle as $Y(1) = b_0 A$ and in general at the n-th clock cycle as

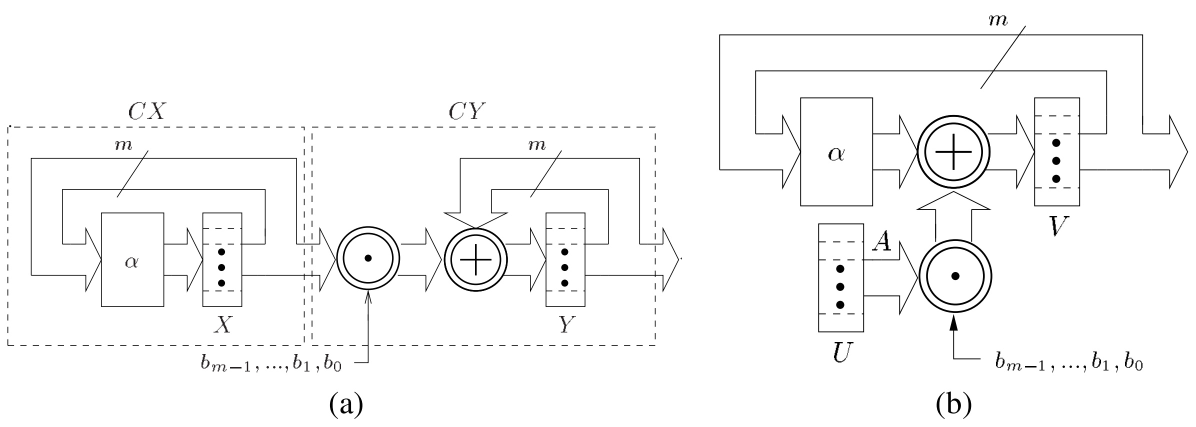


Figure 8: (a) LSB first bit-serial multiplier. (b) MSB first bit-serial multiplier.

$Y(n) = b_0 A + \sum_{i=1}^{n-1} b_i X(i),\ 1 < n \leq m$. Using (6) and noting the fact that $X(n) = X^{(n)}$, one can determine that after m clock cycles Y contains $C = AB \in GF(2^m)$, i.e., $Y(m) = C$. Since the coordinates of B enter the multiplier from the least significant bit (LSB), i.e., b_0, this multiplier is referred to as the LSB first bit-serial multiplier.

5.1.1 Fault Detection Structures

One can also realize Figure 4(a) (and Figure 4(b)) in a bit serial fashion as shown in Figure 9(a) (and Figure 9(b)). Compared to Figure 8(a), one bit is added to the bus of the multiplier and all modules are replaced with the corresponding modules that can detect errors (shown as shaded modules in Figure 9(a)). Also, two m-bit registers $X = (x_{m-1},\ \cdots,\ x_1,\ x_0)$ and $Y = (y_{m-1},\ \cdots,\ y_1,\ y_0)$ are replaced with $m+1$-bit registers $X' = (x_m,\ x_{m-1},\ \cdots,\ x_1,\ x_0)$ and $Y' = (y_m,\ y_{m-1},\ \cdots,\ y_1,\ y_0)$. In Figure 9(a), we assume that X' and Y' are initialized with $X'(0) = (p_A,\ a_{m-1},\ \cdots,\ a_1,\ a_0)$ and $Y'(0) = (0,\ 0,\ \cdots,\ 0,\ 0)$, respectively. Then, after m clock cycles and in a fault free situation, the output of Y' will be $Y'(m) = (\hat{p}_C,\ c_{m-1},\ \cdots,\ c_1,\ c_0)$ and $\hat{e} = \sum_{i=0}^{m-1} c_i + \hat{p}_C = 0$.

If a single fault (permanent or transient) occurs in the α' module and the register X' (except the x_m bit and the XOR gate connected to it) in Figure 9(a), then in each clock cycle the number of erroneous bits of Y' may increase. This is because in the α' module of Figure 9(a), there is a cyclic shift with addition operation (see Figure 9(b) for details) and it may increase the number of errors. Thus, at the end of the m-th clock cycle, there is 50% chance that this fault is detected. If

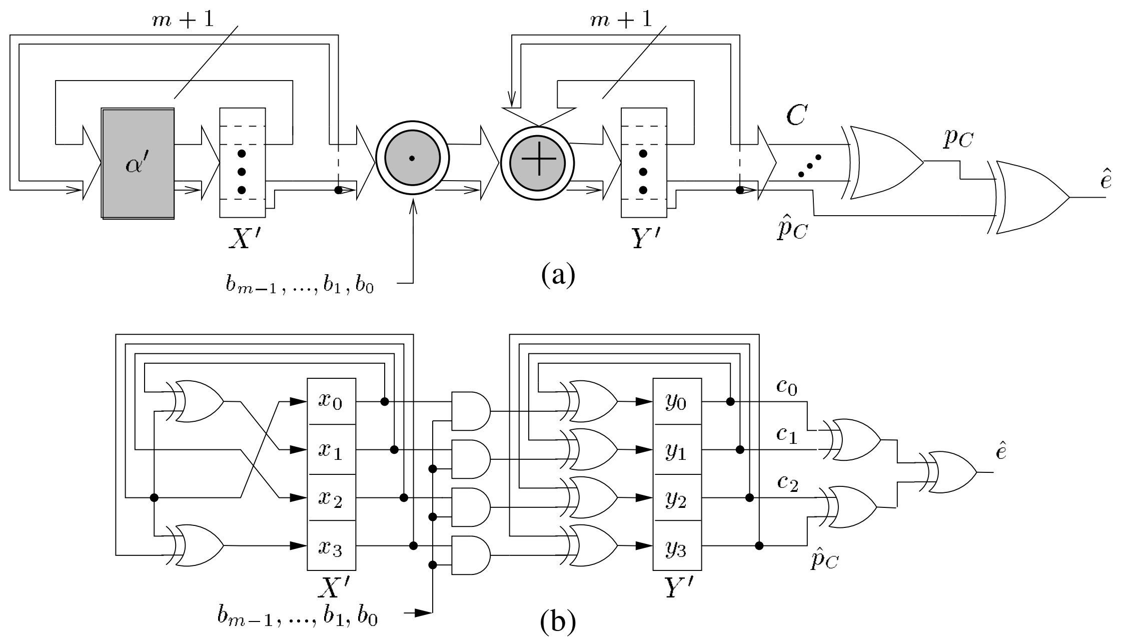


Figure 9: Fault detection of the LSB first bit-serial multiplier (comparison is made after the last clock cycle): (a) the architecture over $GF(2^m)$, (b) an example over $GF(2^3)$.

the single fault occurs in other parts of Figure 9(a) (including input and output of x_m, the output of shaded pass-thru and sum modules as well as Y' and the PG circuit), then at the end of the m-th clock cycle at most one bit of the register Y' will be in error and thus it will be detected by $\hat{e}$. As a result, the total fault coverage of Figure 9 is greater than 50%.

It is noted that we can use (29) to initialize X' and Y' registers with $X'(0) = (0, a_{m-1}, \cdots, a_1, a_0)$ and $Y'(0) = (p_A p_B,\ 0,\ \cdots,\ 0,\ 0)$, respectively and then obtain $Y'(m) = (\hat{p}_C,\ c_{m-1},\ \cdots,\ c_1,\ c_0)$ after m clock cycles. This helps us to increase the probability of fault detection of the multiplier by duplicating the circuits related to x_m and y_m. Thus, by adding one extra gate to shaded modules of Figure 9(a) and replacing X' and Y' with two $m{+}2$-bit registers $X'' = (x_{m+1}, x_m, x_{m-1}, \cdots, x_1, x_0)$ and $Y'' = (y_{m+1},\ y_m,\ y_{m-1},\ \cdots,\ y_1,\ y_0)$, the new multiplier is obtained. In fault free situation, if $X''(0) = (0,\ p_A,\ a_{m-1},\ \cdots,\ a_1,\ a_0)$ and $Y''(0) = (p_A p_B,\ 0,\ 0,\ \cdots,\ 0,\ 0)$, then $Y''(m) = (\hat{p}_C,\ \hat{p}_C,\ c_{m-1},\ \cdots,\ c_1,\ c_0)$ after m clock cycles. Then, in the final clock cycles, last two bits of Y'' (y_{m+1} and y_m) and $\sum_{i=0}^{m-1} y_i$ should be the same as p_C.

Fault Detection in Every Clock Cycle

We can also increase the probability of fault detection in the bit-serial multiplier of Figure 8(a), by checking the contents of two registers in every clock cycle. Consider Figure 8(a) before the triggering of the $n+1$-th clock cycle. At this time, the input and output of the X register are $X(n+1)$ and $X(n)$, respectively and using Lemma 1 for the α module of Figure 8(a), we have $\hat{p}_{X(n+1)} = p_{X(n)} + x_{m-1}(n) = \sum_{i=0}^{m-2} x_i(n)$, where $x_i(n) \in GF(2)$ is the i-th coordinate of $X(n)$. In order to compare $\hat{p}_{X(n)}$ with the actual value of $p_{X(n)}$, we need to store $\hat{p}_{X(n+1)}$ into a 1-bit register D_X as shown in Figure 10. Then, after the n-th clock cycle, $X(n)$ appears at the output of register X and the actual value of $p_{X(n)}$ is evaluated and compared with the value of D_X, i.e., $\hat{p}_{X(n)}$, using the last XOR gate of Figure 10 that generates $\hat{e}_X(n)$. Similar expression can be obtained for the Y register. Since $Y(n+1) = Y(n) + b_n X(n)$, then

$$\hat{p}_{Y(n+1)} = p_{Y(n)} + b_n p_{X(n)}. \tag{44}$$

In Figure 10, $\hat{p}_{Y(n+1)}$ is implemented using (44) and it is delayed by a 1-bit register D_Y to obtain $\hat{p}_{Y(n)}$. Then, it is compared with the actual value of $p_{Y(n)}$ as shown in Figure 10. Therefore, if no faults occur, after the first clock cycle, both $\hat{e}_X(n) = \hat{p}_{X(n)} + p_{X(n)}$ and $\hat{e}_Y(n) = \hat{p}_{Y(n)} + p_{Y(n)}$ should be 0 and remain 0 until the last clock cycle, i.e., $\hat{e}_X(n) = 0$ and $\hat{e}_Y(n) = 0$ for $1 \leq n \leq m$. If either $\hat{e}_X(n)$ or $\hat{e}_Y(n)$ is not 0 during at least one clock cycle, i.e., one value of n for $1 \leq n \leq m$, the existence of a fault is detected. For the cryptographic applications where m is a large number, it is most likely that either $\hat{e}_X(n)$ or $\hat{e}_Y(n)$ is 1 in at least one clock cycle if a single permanent fault occurs. It is noted that the probability of fault detection and hence the fault coverage may be decreased if a single transient fault occurs.

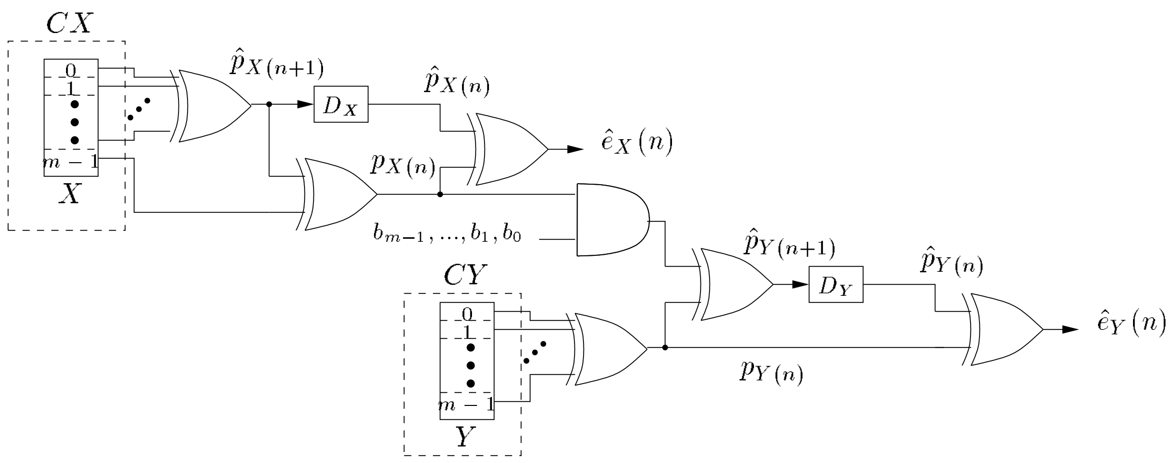


Figure 10: Fault detection in the bit-serial multiplier of Figure 8(a) (comparisons are made in every clock cycle after the first clock cycle).

5.2 MSB First Multiplier

The PB multiplication of A and B in (5) can also be written as

$$C = (((b_{m-1}A\alpha + b_{m-2}A)\alpha + b_{m-3}A) + \cdots + b_1A)\alpha + b_0A. \tag{45}$$

Equation (45) can be realized by an architecture as shown in Figure 8(b). In this architecture, which is known as the most significant bit (MSB) first bit-serial multiplier, the registers U and V are initialized with $A = (a_{m-1}, \cdots, a_1, a_0)$ and $0 = (0, \cdots, 0, 0)$, respectively. Let $V(n)$ denote the content of $V = (v_{m-1}, \cdots, v_1, v_0)$ at the n-th, $0 \leq n \leq m$ clock cycle. Then, one can verify that $V(0) = 0$, $V(1) = b_{m-1}A$, $V(2) = b_{m-1}A\alpha + b_{m-2}A, \cdots,$

$$V(n+1) = V(n)\alpha + b_{m-n-1}A,\ 0 \leq n \leq m-1. \tag{46}$$

Thus, after the m-th clock cycle $V(m) = C$ (refer to (45)), i.e., the register V contains the coordinates of C.

5.2.1 Fault Detection Structures

In order to detect a single fault in the architecture of Figure 8(b), we can use the linear property of parity prediction for equation (46) to obtain

$$\hat{p}_{V(n+1)} = \hat{p}_{V(n)\alpha} + b_{m-n-1}p_A. \tag{47}$$

Using (16), we can obtain $\hat{p}_{V(n)\alpha} = p_{V(n)} + v_{m-1}(n)$ and substituting into (47) results in

$$\hat{p}_{V(n+1)} = p_{V(n)} + v_{m-1}(n) + b_{m-n-1}p_A, \tag{48}$$

where $v_{m-1}(n) \in GF(2)$ denote bit $m-1$ of V, i.e., v_{m-1}, at the n-th clock cycle. Equation (47) is used to obtain $\hat{p}_C$ by replacing the m-bit register $V = (v_{m-1}, \cdots, v_1, v_0)$ in Figure 8(b) by the $m+1$-bit register $V' = (v_m, v_{m-1}, \cdots, v_1, v_0)$, where $v_m(n)$ is used to store $p_{V(n)}$. Since the initial value of V is 0, i.e., $V(0) = (0, \cdots, 0, 0) \in GF(2^m)$, then $v_m(0) = p_{V(0)} = 0 \in GF(2)$. Thus, the circuit of Figure 11(a) implements (47), where U' and V' are two $m+1$-bit registers and should be initialized with $U'(0) = (p_A, a_{m-1}, \cdots, a_1, a_0)$ and $V'(0) = (0, 0, \cdots, 0, 0)$, respectively. After the m-th clock cycle, V' will have $V'(m) = (\hat{p}_C, c_{m-1}, \cdots, c_1, c_0)$ and its m-th bit, i.e., $v_m(m) = \hat{p}_C$ is compared with $\sum_{i=0}^{m-1} v_i(m) = p_C$ as shown in Figure 11(a). The corresponding fault detection multiplier over $GF(2^3)$ generated by $z^3 + z + 1$ is shown in Figure 11(b).

In Figure 11, since a cyclic shift in the α' module is in the main loop of the multiplier, any single fault (permanent or transient) will result in more than one error in V' at the final clock cycle. Thus, unlike the structure of Figure 9 which has the fault coverage greater than 50%, the fault coverage of Figure 11 is 50%.

Fault Detection in Every Clock Cycle

A single fault (permanent or transient) may generate an even number of errors that cannot be detected by the MSB-first multiplier of Figure 11. To overcome this problem, similar to the LSB-

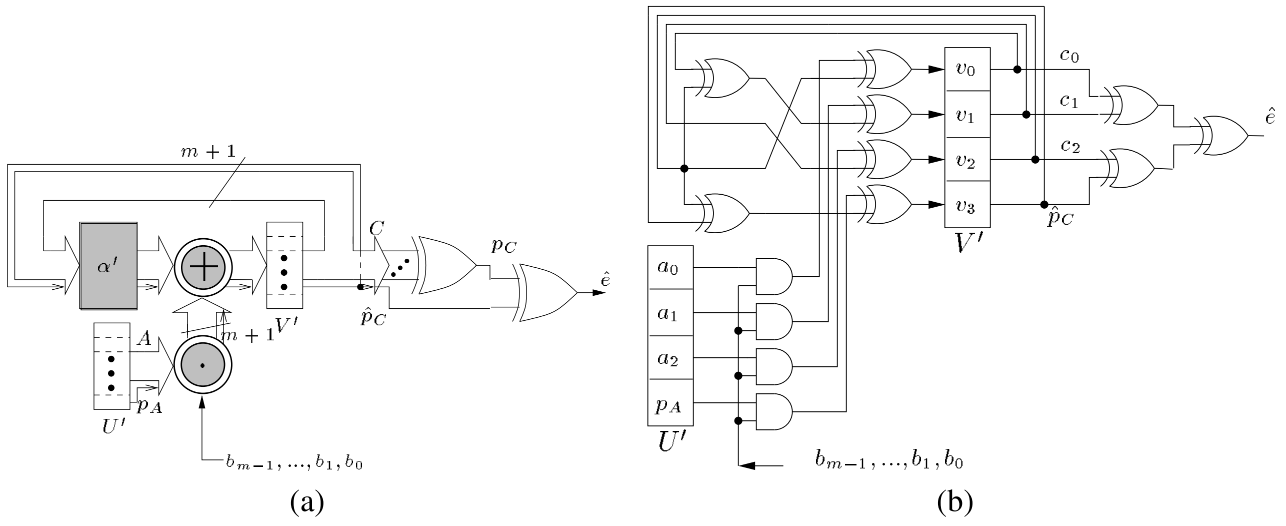


Figure 11: Fault detection of the MSB first bit-serial multiplier: (a) the architecture over $GF(2^m)$, (b) an example over $GF(2^3)$.

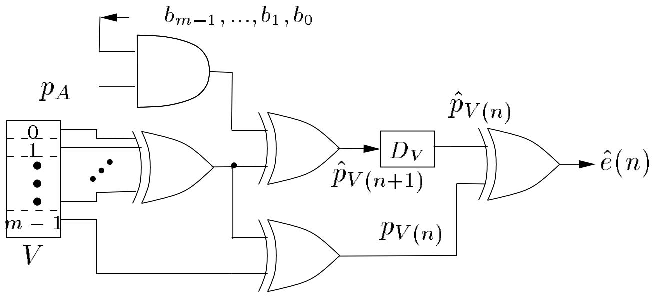


Figure 12: Fault detection in the bit-serial multiplier of Figure 8(b) (comparison is made in every clock cycle after the first clock cycle).

first multiplier, the content of V is checked in each clock cycle as shown in Figure 12. In this figure, $p_{V(n)} = \sum_{i=0}^{m-1} v_i(n)$ is compared with $\hat{p}_{V(n)}$ by using an XOR gate that generates $\hat{e}(n)$. As seen in the figure, $\hat{p}_{V(n)}$ is obtained from $\hat{p}_{V(n+1)}$ using the 1-bit register D_V and $\hat{p}_{V(n+1)}$ is implemented as $\sum_{i=0}^{m-2} v_i(n) + b_{m-n-1}p_A$ (see (47)). Thus, if no faults occur, $\hat{e}(n)$ should be 0 after the first clock cycle until the m-th clock cycle, i.e., $\hat{e}(n) = 0$ for any $1 \le n \le m$. If it flags $\hat{e}(n) = 1$ during at least one clock cycle, say at the n'-th clock cycle, it implies that equation (48) does not hold for $n = n'$ and a fault is detected by $\hat{e}(n') = 1$. It is noted that the fault coverage may be reduced if a transient fault is applied.

6 Discussion and Conclusions

In this article, we have considered detection of single stuck-at faults in polynomial basis multipliers. Compared to previously published results [5], the work presented here is quite generic in the sense that it can be applied to any irreducible polynomial defining the field. The area overhead cost of the fault detection structures is much lower than having an extra multiplier as needed in the dual modular redundant system.

Table 1 shows the complexities and overheads of the fault detection structures in terms of area and time. The complexities shown in the table consist of the complexities of the original multipliers and their corresponding overheads, which are given in the square brackets. In this table, T_A and T_X are delays of an AND gate and an XOR gate, respectively. Also, $T_{LCBP} = T_A + (2(\omega - 2) + \lceil \log_2(m-1) \rceil)T_X$, is the time delay of Figure 6(a), where ω is the number of non-zero terms of the irreducible polynomial. Let $T_{LSB} \geq T_A + T_X$ and $T_{MSB} \geq T_A + T_X$ be the clock periods of the original multipliers of Figure 8(a) and Figure 8(b), respectively. Then, T_{LSB} and T_{MSB} will be the clock periods of Figure 9(a) and 11(a), respectively, since the overhead will not affect on the clock periods. However, this is not the case in Figures 10 and 12. Let us denote the clock periods of the corresponding multipliers of Figure 10 and Figure 12 by T'_{LSB} and T'_{MSB}, respectively. From these figures, one can see that $T'_{LSB} \geq T_A + (2 + \lceil \log_2(m-1) \rceil)T_X$ and $T'_{MSB} \geq (1 + \lceil \log_2(m-1) \rceil)T_X$. In the last two columns of the table, the overhead percentages and fault coverages are given. The overhead values are obtained for one finite field recommended by NIST for ECDSA. It is interesting to note that while the proposed fault detecting bit-parallel multipliers have a space complexity of $O(m^2)$, their overhead cost is $O(m)$.

Although our discussions have centered around bit-parallel and bit-serial multipliers over $GF(2^m)$, by combining the error-control schemes for serial and parallel multipliers, one can develop fault detecting hybrid multipliers over composite fields [12].

Fault detection muliplier	Complexities=original+[overhead]		Overhead%[a]	Coverage
Traditional Bit-Parallel (Fig. 4a or Fig. 5a)	#AND	$m^2 + [m]$	0.6%	100%
	#XOR	$(m+\omega-2)(m-1)+[4m-2]$	2.4%	
	Delay	$T_A + mT_X + [\lceil \log_2(m+1) \rceil T_X]$	4.9%[b]	
Low Complexity Bit-Parallel (Fig. 7a)	#AND	$m^2 + [1]$	0.004%	100%
	#XOR	$(m+\omega-2)(m-1)+[3m]$	1.8%	
	Delay	$T_{LCBP} + [(1 + \lceil \log_2 m \rceil) T_X]$	69.2%	
LSB-First (Fig. 9a, comparison is made in the last clock)	#AND	$m + [1]$	0.6%	>50%[g]
	#XOR	$m+\omega-2+[m+2]$	99%	
	#1-bit reg.	$2m + [2]$	0.6%	
	Delay	$mT_{LSB} + [\lceil \log_2(m+1) \rceil T_X]$	$\leq 2.4\%$	
LSB-First (Fig. 10, comparison is made in every clock)	#AND	$m + [1]$	0.6%	100%[h]
	#XOR	$m+\omega-2+[2m+1]$	197%	
	#1-bit reg.	$2m + [2]$	0.6%	
	Delay	$(m+1)T'_{LSB}$ [c]	0.6%-450%[d]	
MSB-First (Fig. 11a, comparison is made in the last clock)	#AND	$m + [1]$	0.6%	50%
	#XOR	$m+\omega-2+[m+2]$	99%	
	#1-bit reg.	$2m + [2]$	0.6%	
	Delay	$mT_{MSB} + [\lceil \log_2(m+1) \rceil T_X]$	$\leq 2.4\%$	
MSB-First (Fig. 12, comparison is made in every clock)	#AND	$m + 1$	0.6%	100%[h]
	#XOR	$m+\omega-2+[m+1]$	99%	
	#1-bit reg.	$2m + [1]$	0.3%	
	Delay	$(m+1)T'_{MSB}$ [e]	0.6%-350%[f]	

Table 1: Complexities and overhead cost of fault detection structures for multiplication using polynomial bases. Note that the fault coverage is for both permanent and transient faults unless stated.

[a]Overhead%=100×overhead/original, where $m = 163$, $\omega = 5$ and we assume that $T_A = T_X$.

[b]If the array of sum modules in the traditional multiplier is implemented using binary trees of XOR gates, then this value will be much higher.

[c]$(m+1)T'_{LSB} = mT_{LSB} + [(m+1)T'_{LSB} - mT_{LSB}]$.

[d]0.6% is obtained if $T_{LSB} = T'_{LSB}$ and 450% is obtained if $T_{LSB} = T_A + T_X$ and $T'_{LSB} = T_A + (2 + \lceil \log_2(m-1) \rceil)T_X$.

[e]$(m+1)T'_{MSB} = mT_{MSB} + [(m+1)T'_{MSB} - mT_{MSB}]$.

[f]0.6% is obtained if $T_{MSB} = T'_{MSB}$ and 350% is obtained if $T_{MSB} = T_A + T_X$ and $T'_{MSB} = (1 + \lceil \log_2(m-1) \rceil)T_X$.

[g]The fault coverage will be increased if two parity checks, as mentioned earlier, are used.

[h]The fault coverage may be reduced if a transient fault is applied.

Acknowledgments

The authors would also like to thank the reviewers for their comments. A preliminary version of this article was presented at Workshop on Cryptographic Hardware and Embedded Systems 2002 (CHES 2002) [13]. This work has been supported in part by an NSERC Discovery grant awarded to A. Reyhani-Masoleh and in part by NSERC Discovery and Strategic grants awarded to M. A. Hasan.

References

[1] G. B. Agnew, T. Beth, R. C. Mullin, and S. A. Vanstone. "Arithmetic Operations in $GF(2^m)$". *Journal of Cryptology*, 6:3–13, 1993.

[2] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri. "Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard". *IEEE Transactions on Computers, Special Issue on Cryptographic Hardware and Embedded Systems*, 52(4):492–505, April 2003.

[3] D. Boneh, R. A. DeMillo, and R. J. Lipton. "On the Importance of Eliminating Errors in Cryptographic Computations". *Journal of Cryptology*, 14:101–119, 2001.

[4] M. Ciet and M. Joye. "Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults". *Designs, Codes and Cryptography*, 36(1):33–43, 2005.

[5] S. Fenn, M. Gossel, M. Benaissa, and D. Taylor. "On-Line Error Detection for Bit-Serial Multipliers in $GF(2^m)$". *Journal of Electronic Testing: Theory and Applications*, 13:29–40, 1998.

[6] A. Halbutogullari and C. K. Koç. "Mastrovito Multiplier for General Irreducible Polynomials". *IEEE Transactions on Computers*, 49(5):503–518, May 2000.

[7] B. W. Johnson. *Design and Analysis of Fault-Tolerant Digital Systems*. Addison-Wesley Publishing Company, 1989.

[8] M. Joye, A. K. Lenstra, and J. J. Quisquater. "Chinese Remaindering Based Cryptosystems in the Presence of Faults". *Journal of Cryptology*, 12:241–245, 1999.

[9] E. D. Mastrovito. "VLSI Designs for Multiplication over Finite Fields $GF(2^m)$". In *LNCS-357, Proc. AAECC-6*, pages 297–309, Rome, July 1988. Springer-Verlag.

[10] E. D. Mastrovito. *VLSI Architectures for Computation in Galois Fields*. PhD thesis, Linkoping Univ., Linkoping Sweden, 1991.

[11] M. Nicolaidis, R. O. Duarte, S. Manich, and J. Figueras. "Fault-Secure Parity Prediction Arithmetic Operators". *IEEE Design and Test of Computers*, pages 60–71, April-June 1997.

[12] C. Paar, P. Fleishmann, and P. Soria-Rodriguez. "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents". *IEEE Transactions on Computers*, 48(10):1025–1034, Oct. 1999.

[13] A. Reyhani-Masoleh and M. A. Hasan. “Error Detection in Polynomial Basis Multipliers over Binary Extension Fields”. In *Cryptographic Hardware and Embedded Systems - CHES 2002, LNCS No. 2523, Springer Verlag, Berlin, Germany, 2002*, pages 515–528, August 2002.

[14] A. Reyhani-Masoleh and M. A. Hasan. “Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over $GF(2^m)$”. *IEEE Transactions on Computers*, 53(8):945–959, August 2004.

[15] F. Rodriguez-Henriquez and C. K. Koç. “Parallel Multipliers Based on Special Irreducible Pentanomials”. *IEEE Transactions on Computers*, 52(12):1535–1542, December 2003.

[16] Gadiel Seroussi. “Table of Low-Weight Binary Irreducible Polynomials”. *HP Labs Tech. Report HPL-98-135*, August 1998.

[17] H. Wu. “Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis”. *IEEE Transactions on Computers*, 51(7):750–758, July 2002.

[18] H. Wu and M. A. Hasan. “Efficient Exponentiation of a Primitive Root in $GF(2^m)$”. *IEEE Transactions on Computers*, 46(2):162–172, Feb. 1997.

[19] K. Wu, R. Karri, G. Kuznetsov, and M. Goessel. Low Cost Concurrent Error Detection for the Advanced Encryption Standard. In *Proceedings of the IEEE International Test Conference (ITC 2004)*, pages 1242–1248, 2004.

[20] T. Zhang and K. K. Parhi. “Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials”. *IEEE Transactions on Computers*, 50(7):734–748, July 2001.