Towards Fault-Tolerant Cryptographic Computations over Finite Fields

ARASH REYHANI-MASOLEH and M. ANWAR HASAN
University of Waterloo

Cryptographic schemes, such as authentication, confidentiality, and integrity, rely on computations in very large finite fields, whose hardware realization may require millions of logic gates. In a straightforward design, even a single fault in such a complex circuit is likely to yield an incorrect result and may be exploited by an attacker to break the cryptosystem. In this regard, we consider computing over finite fields in presence of certain faults in multiplier circuits. Our work reported here deals with errors caused by such faults in polynomial basis multipliers over finite fields of characteristic two and presents a scheme to correct single errors. Towards this, pertinent theoretical results are derived, and both bit-parallel and bit-serial fault tolerant multipliers are proposed.

Categories and Subject Descriptors: B.2.4 [**Hardware**]: High-Speed Arithmetic; B.8.1 [**Hardware**]: Reliability, Testing, and Fault-Tolerance

General Terms: Algorithms, Security

Additional Key Words and Phrases: Error correction, fault-tolerant computing, finite fields, polynomial basis multiplier, security

1. INTRODUCTION

Arithmetic operations in finite fields are extensively used in cryptography and coding [Lidl and Niederreiter 1994; Menezes et al. 1993], and among them multiplication is the main operation [Mastrovito 1991; Halbutogullari and Koc 2000; Zhang and Parhi 2001]. In cryptographic applications, the field size can be very large (say, 2^{1024}). When VLSI technologies are used to implement a processor, which can perform operations over a large field, millions of logic gates may be needed. It is a formidable task to implement a processor for large fields, which will be fault free.

There are two basic types of faults: permanent faults and transient faults. Sophisticated testing schemes can identify permanent faults, but not transient faults that commonly exist in integrated circuit [Pradhan 1996; Johnson 1989].

Authors' addresses: A. Reyhani-Masoleh, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1; email: arash@secure2.uwaterloo.ca; M. A. Hasan, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1; email: ahasan@ece.uwaterloo.ca.

The latter has motivated the development of fault-tolerant techniques, which can tolerate all permanent and transient faults during the system operation. Generally speaking, fault-tolerant arithmetic can be obtained by using time and space redundancy. The time redundancy approach utilizes only one module to provide error correction by repeating the operation at least three times. It requires a 200% overhead in time, which is not appropriate or desirable for many systems. Space redundancy uses coding techniques or modular redundant schemes with voting to perform the corrective action. The most popular modular redundant schemes is the triple modular redundancy (TMR), which produces the correct result despite any existing fault in one of the three modules. The space overhead of the TMR is at least 200%. In coding techniques, the correction process should be fault free or at least self-checking [Rao and Fujiwara 1989].

A relatively less complex approach is to have error *detection* capability only. Towards this, a few finite field arithmetic units have been reported in the literature [see, e.g., Fenn et al. 1998; Reyhani-Masoleh and Hasan 2002]. While error detection may be adequate for some applications, error correction can potentially offer certain crucial advantages, such as higher yield factors and increased availability. In this article, we present multipliers for fields $GF(2^m)$ whose operations are resistant to errors caused by certain faults. We use the most widely used *polynomial* basis to represent the field elements and develop an error correction scheme, which can be applied to both bit-parallel and bit-serial type multipliers. To the best of our knowledge, no previous article has addressed this issue of correcting errors in the finite field multiplier.

For the purpose of this investigation, first we investigate the multiplier circuit with single faults. Such a single-fault scenario simplifies our analysis. This fault is modeled as a stuck-at fault, which appears to be the most common model used for logical faults. For this model, a fault in a logical gate 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 [Lala 1985]. In the sequel, we also show that certain multiple faults, which produce a single-bit error at the output, are also tolerated by the proposed multiplier structures.

The organization of this article is as follows. In Section 2, an architecture of multiplication using the widely used polynomial basis is discussed and fault-tolerent strategies are outlined. In Section 3, an error correcting code is proposed and it is applied to the so-called α module of the multiplier. Error correction in the other multiplier modules are considered in Section 4. Then Section 5 puts together the fault-tolerent modules to yield a single error-tolerant bit-parallel and bit-serial multiplier structures. Finally, Section 6 gives a few concluding remarks.

2. REVIEW OF POLYNOMIAL BASIS MULTIPLICATION

In this section, we review polynomial basis multiplication over $GF(2^m)$ and its basic bit-parallel structure. Later in this paper, we will investigate error correction schemes for this multiplier structure.

Let us construct the finite field $GF(2^m)$ using polynomial basis $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$ as

$$GF(2^m) = \left\{ A = \sum_{i=0}^{m-1} a_i \alpha^i | a_i \in GF(2),\ 0 \le i \le m-1 \right\} \tag{1}$$

where $\alpha \in GF(2^m)$ is a root of an irreducible polynomial

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

of degree m over $GF(2)$. In (2), ω is the Hamming weight of $F(z)$ and ρ_j's are integers such that $1 \le \rho_1 < \rho_2 < \rho_3 < \cdots < \rho_{\omega-2} \le m-1$. For convenience, each element of $A = \sum_{i=0}^{m-1} a_i \alpha^i \in GF(2^m),\ a_i \in \{0,\ 1\}$, can be written in vector notation as

$$\underline{a} = [a_0,\ a_1,\ a_2, \ldots, a_{m-1}] \tag{3}$$

where a_i's are the coordinates of A with respect to polynomial basis.

Let C be the product of two field elements A and $B = \sum_{i=0}^{m-1} b_i \alpha^i$ of $GF(2^m)$. Then

$$C = A \cdot B \mod F(\alpha) = \sum_{i=0}^{m-1} b_i \cdot X^{(i)} \tag{4}$$

where

$$X^{(i)} = \alpha \cdot X^{(i-1)} \mod F(\alpha), \qquad 1 \le i \le m-1 \tag{5}$$

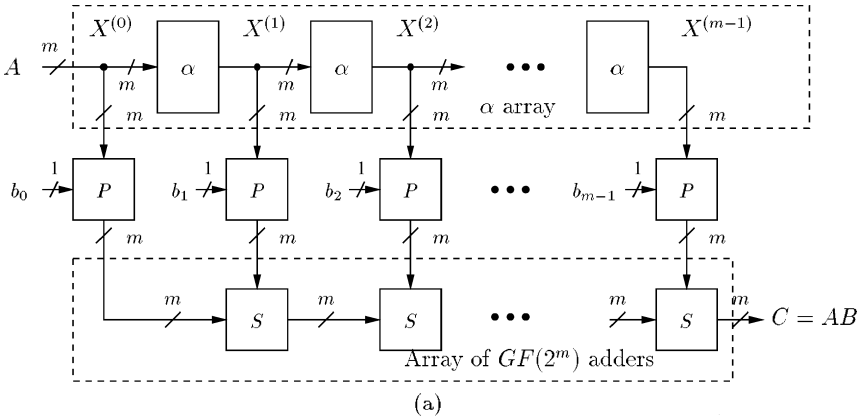
and $X^{(0)} = A$.

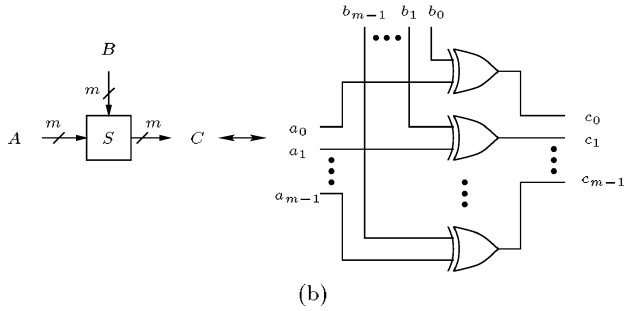
Based on equation (4), various structures for finite field multipliers are possible. One bit-parallel structure is shown in Figure 1(a), which consists of three types of modules referred to as *sum* (S), *pass-thru* (P), and α modules [Reyhani-Masoleh and Hasan 2002]. Each sum module is to add two $GF(2^m)$ elements while a pass-thru module, depending on its one-bit control (horizontal) input, generates $0 \in GF(2^m)$ or reproduces its input. The α module multiplies its input by α and reduces the result modulo $F(\alpha)$. Denote the input and the output of the α module as U and V respectively. Let us define the set $\Gamma = \{k | k = \rho_j,\ 1 \le j \le \omega-2\}$ and its complement $\Gamma' = \{k | 0 \le k \le m-1,\ k \ne \rho_j,\ 1 \le j \le \omega-2\}$, where Γ and Γ' correspond to the non-zero and zero coefficients of $F(z)$, respectively. Then, one can write

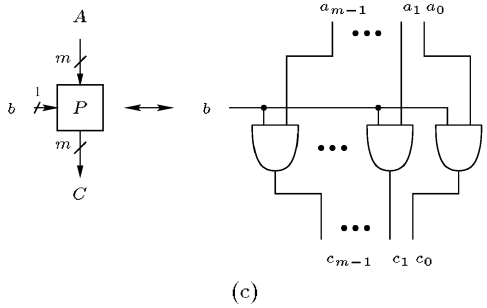
$$\begin{aligned} V &= U \cdot \alpha \bmod F(\alpha) = \sum_{i=0}^{m-1} u_i \alpha^{i+1} \bmod F(\alpha) = \sum_{i=1}^{m-1} u_{i-1} \alpha^i + u_{m-1} \alpha^m \bmod F(\alpha) \\ &= u_{m-1} + \sum_{i \in \Gamma} (u_{i-1} + u_{m-1}) \alpha^i + \sum_{i \in \Gamma'} u_{i-1} \alpha^i \end{aligned} \tag{6}$$

Thus from (6), the coordinates of V can be obtained as

$$v_i = \begin{cases} u_{\rho_j - 1} + u_{m-1} & i = \rho_j, \qquad 1 \le j \le \omega - 2, \\ u_{i-1 \bmod m} & \text{otherwise.} \end{cases} \tag{7}$$







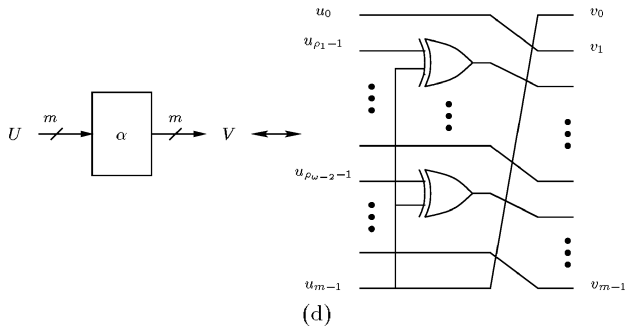


Fig. 1. (a) Multiplication of two elements in $GF(2^m)$. (b–d) Details of the sum, pass-thru, and α modules, respectively.

Figures 1(b)-(d) show hardware implementations of the abovementioned sum, pass-thru, and α modules using two-input logic gates. From these diagrams one can see that each sum, pass-thru, and α module requires m XOR gates, m AND gates, and $(\omega - 2)$ XOR gates, respectively. 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 of the minimum Hamming weight (i.e., a trinomial). Similarly, the complexity is maximum when $F(z)$ is of the maximum Hamming weight (i.e., an all-one polynomial).

In the following sections, we investigate error correction schemes for $GF(2^m)$ multiplication operation that relies on the architecture shown in Figure 1(a). Towards this effort, first the three modules of the multiplier structure are considered, one at a time, for correcting single errors in them. This requires modification of the modules mainly by adding extra hardware to them. Then the error-resistant modules are combined, at the expense of further hardware, to develop an architecture for the entire multiplier that can operate and produce correct results at the presence of an error caused by one or more faults in the multiplier circuit.

3. FAULT TOLERANT α MODULE

In this section, we introduce a new single error correcting code whose generator matrix is obtained by using the property of the α module. Later this code will also be used for other modules of the multiplier.

3.1 Modeling and Encoding of Faulty α Module Output

In the fault-free situation, the relationship between the input and the output of the α module is given by (6). Using the vector notation in (3), equation (6) can also be written as

$$\underline{v} = \underline{u} \cdot \mathbf{M} \tag{8}$$

where

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ 1 & f_1 & f_2 & \cdots & f_{m-1} \end{bmatrix} \triangleq \left[\underline{\mu}_0^{\mathrm{T}} \; \underline{\mu}_1^{\mathrm{T}} \; \cdots \; \underline{\mu}_{m-1}^{\mathrm{T}} \right] \tag{9}$$

and $\underline{\mu}_j^{\mathrm{T}}$, $0 \leq j \leq m-1$, is column j of $\mathbf{M}$. In (9), f_i is 1 if $i \in \Gamma$ and 0 otherwise. Note that a stuck-at fault in one of the $(\omega - 2)$ XOR gates of the α module causes at most one error in the output bits.

For $j \in \Gamma$, assume that the jth gate in Figure 1(d) is faulty. Then all the output coordinates, except v_j, are error free. If the upper input of the jth gate is stuck, then the erroneous jth coordinate is

$$\hat{v}_j = \begin{cases} u_{m-1} & \text{for s-a-0} \\ \overline{u}_{m-1} & \text{for s-a-1} \end{cases} \tag{10}$$

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

$$\hat{v}_j = \begin{cases} u_{j-1} & \text{for s-a-0} \\ \overline{u}_{j-1} & \text{for s-a-1.} \end{cases} \tag{11}$$

In order to correct a single error in $\underline{v}$ (which is the vector representing the output of the α module), a number of parity bits are needed. Let us assume that there are r parity bits denoted as $y_0, y_1, \ldots, y_{r-1}$, and the corresponding vector is $\underline{y} = [y_0, y_1, \ldots, y_{r-1}]$. This parity vector is combined with $\underline{v}$ to obtain a codeword $\underline{w} = [\underline{v}\ \underline{y}]$.

Let $\mathbf{R}$ be an $m \times r$ *parity* matrix that generates the parity vector from the α module's input, that is,

$$\underline{y} = \underline{u} \cdot \mathbf{R}. \tag{12}$$

Then the codeword can be written as $\underline{w} = \underline{u} \cdot \mathbf{G}$, where the $m \times (m+r)$ *generator* matrix $\mathbf{G}$ is defined as

$$\mathbf{G} = [\mathbf{M}\ \mathbf{R}]. \tag{13}$$

By permuting the columns of $\mathbf{G}$, one can obtain a more conventional form of the generator matrix as follows:

$$\widetilde{\mathbf{G}} = [\mathbf{I_m}\ \mathbf{B}] \tag{14}$$

where $\mathbf{I_m}$ is the $m \times m$ unity matrix and $\mathbf{B}$ is an $m \times r$ matrix. Below, a method for obtaining $\widetilde{\mathbf{G}}$ from $\mathbf{G}$, that is, $\mathbf{B}$ from $\mathbf{M}$ and $\mathbf{R}$, is given.

Let $\mathbf{H}$ and $\widetilde{\mathbf{H}}$ be the *parity check* matrices corresponding to $\mathbf{G}$ and $\widetilde{\mathbf{G}}$, respectively. From the property of parity check matrices that every row vector of $\widetilde{\mathbf{G}}$ is orthogonal to every row vector of $\widetilde{\mathbf{H}}$ [Lin and Costello 1983; Vanstone and van Oorschot 1989], one has

$$\widetilde{\mathbf{G}} \cdot \widetilde{\mathbf{H}}^{\mathrm{T}} = 0. \tag{15}$$

Combining (14) and (15), the parity check matrix $\widetilde{\mathbf{H}}$ can be written as

$$\widetilde{\mathbf{H}} = [\mathbf{B}^{\mathrm{T}}\ \mathbf{I_r}]. \tag{16}$$

If $\mathbf{B}$ is chosen such that no two columns of $\widetilde{\mathbf{H}}$ are identical, then $\widetilde{\mathbf{H}}$ is a single error correcting parity check matrix [Lin and Costello 1983]. One can obtain the corresponding $\mathbf{H}$ by applying the same permutation used for obtaining $\mathbf{G}$ from $\widetilde{\mathbf{G}}$.

Below, we discuss how to determine the parity matrix $\mathbf{R}$, which will enable us to correct single errors of the α module output.

3.2 Parity Matrix $\mathbf{R}$

Using (2), we introduce another notation $\underline{e}_{\rho_j}^{\mathrm{T}}$, $1 \le j \le \omega - 2$, which is a unity column vector of order $(m-1)$ having an entry of one in position ρ_j and zeros in the other $(m-2)$ entries. For example $\underline{e}_3 = [0, 0, 1, 0, \ldots, 0]$. Using such

unity vectors, we can write the $m \times m$ unity matrix as

$$\mathbf{I_m} = \begin{bmatrix} \underline{e}_1^{\mathrm{T}} & \underline{e}_2^{\mathrm{T}} & \cdots & \underline{e}_{m-1}^{\mathrm{T}} & \underline{\mu}_0^{\mathrm{T}} \\ 0 & 0 & \cdots & 0 & \end{bmatrix}$$

where $\underline{\mu}_0^{\mathrm{T}}$ is column 0 of $\mathbf{M}$.

Note that $\mathbf{G} = [\mathbf{M}\ \mathbf{R}]$ is to be transformed to $\widetilde{\mathbf{G}} = [\mathbf{I_m}\ \mathbf{B}]$ by column permutation and $\mathbf{M}$ (refer to (9)) has $\underline{\mu}_j^{\mathrm{T}} = [\begin{smallmatrix} \underline{e}_j^{\mathrm{T}} \\ 0 \end{smallmatrix}]$ only if $f_j = 0$, for $1 \le j \le m-1$. Thus the parity matrix $\mathbf{R}$ can have the following form

$$\mathbf{R} = [\mathbf{E}_0\ \mathbf{R}'] \tag{17}$$

where $\mathbf{E}_0$ is an $m \times (\omega - 2)$ matrix given as follows:

$$\mathbf{E}_0 = \begin{bmatrix} \underline{e}_{\rho_1}^{\mathrm{T}} & \underline{e}_{\rho_2}^{\mathrm{T}} & \cdots & \underline{e}_{\rho_{\omega-2}}^{\mathrm{T}} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{18}$$

and $\mathbf{R}'$ is an $m \times (r - \omega + 2)$ matrix that is to be determined such that $\mathbf{G}$ is the generator matrix of a single error correcting code.

By substituting (9) and (17) in (13) and then permuting columns to get $\widetilde{\mathbf{G}}$, we have

$$\mathbf{B} = [\mathbf{E}_1\ \mathbf{R}'] \tag{19}$$

where $\mathbf{E}_1$ is an $m \times (\omega - 2)$ matrix, which is similar to $\mathbf{E}_0$ except that it has all one's in the last row, that is,

$$\mathbf{E}_1 = \begin{bmatrix} \underline{e}_{\rho_1}^{\mathrm{T}} & \underline{e}_{\rho_2}^{\mathrm{T}} & \cdots & \underline{e}_{\rho_{\omega-2}}^{\mathrm{T}} \\ 1 & 1 & \cdots & 1 \end{bmatrix}. \tag{20}$$

3.3 Properties of Parity Matrix

For a single error correcting codes, we have to find an $\mathbf{R}'$ such that $\widetilde{\mathbf{H}}$ does not have two identical columns. Depending on the values of m and ω, there are more than one $\mathbf{R}'$ (and hence $\mathbf{R}$) that one can choose to satisfy this condition. Towards this, we have the following lemma whose proof is given in Appendix A.

LEMMA 3.1. *The total number of single error correcting parity check matrices* $\mathbf{H}$ *(and hence* $\mathbf{R}$*) is*

$$N = \begin{cases} 2^n(2^n-1)^{\omega-2} \prod_{j=1}^{m-\omega+1}(2^n-(n+j)), & 3 < \omega \le m \\ (2^n-1)\,(2^n-2) \prod_{j=1}^{m-2}(2^n-(n+j)), & \omega = 3 \\ 2^n(2^n-1)^{m-1}, & \omega = m+1 \end{cases} \tag{21}$$

where $n = r - (\omega - 2)$ *and* ω *is the Hamming weight of* $F(z)$*.*

In (21), the parameter n refers to the number of columns in $\mathbf{R}'$. The value of n determines the number of parity bits, since

$$r = n + (\omega - 2). \tag{22}$$

For a given $F(z)$, by reducing n, we can reduce r. Towards this effect, we have the following theorem.

THEOREM 3.2. *The lower bound on the number of columns in* $\mathbf{R}'$ *is the smallest nonzero positive integer* n *that satisfies the following inequality*

$$2^n > n + m - \omega + 1. \tag{23}$$

PROOF. From Lemma 3.1, N is minimum when n is minimum. Since N is a nonzero positive integer, one must have $N \geq 1$. Then, from (21) one can write
For $3 < \omega \leq m$:

$$\left(2^n(2^n-1)^{\omega-2}\prod_{j=1}^{m-\omega+1}(2^n-(n+j))\right) \geq 1$$

where each factor of the LHS must be nonzero. Thus, for $j = 1, 2, \ldots, m-\omega+1$, one must have $2^n - (n + j)$, that is,

$$2^n - (n + m - \omega + 1) > 0. \tag{24}$$

For $\omega = 3$: like the previous case $((2^n - 1)\,(2^n - 2)\prod_{j=1}^{m-2}(2^n - (n + j))) \geq 1$ implies that

$$2^n - (n + m - 2) > 0. \tag{25}$$

For $\omega = m + 1 : 2^n(2^n - 1)^{m-1} \geq 1$, that is,

$$2^n - 1 > 0. \tag{26}$$

For any nonzero positive n, (26) is always true. Combining (24) and (25), we have

$$2^n - (n + m - \omega + 1) > 0 \qquad \text{for } 3 \leq \omega \leq m. \tag{27}$$

Since the solution to (27) for $\omega = m + 1$ is also the solution to (26), so (27) can also be used for $\omega = m + 1$ and the proof is complete. □

For the two special classes, namely trinomials ($\omega = 3$) and pentanomials ($\omega = 5$), one can apply Theorem 3.2 to obtain the minimum value of n (hence minimum of r) for a given m. This is shown in Table I. Also, when $F(z)$ is an all-one polynomial, the minimum value of n is one and hence the lower bound on the number of parity bits needed is m. This can also be seen in the third row of Table I, where the trinomial for $m = 2$, that is, $z^2 + z + 1$, and pentanomial for $m = 4$, that is, $z^4 + z^3 + z^2 + z + 1$, are also all-one polynomials.

Below, we present an example to illustrate the formulation of a parity matrix using the above results.

Example 1. Let $F(z)$ be $z^4 + z + 1$, which is an irreducible polynomial over $GF(2)$, and let $\alpha \in GF(2^4)$ be its root. For this $F(z)$, we have $m = 4$ and $\omega = 3$. By using (23) and (22), the lower bound of n and number of parity bits are 3

Table I. Minimum Parity Bits in Terms of m for Trinomials and Pentanomials

Trinomials $\omega = 3$			Pentanomials $\omega = 5$		
m	min. n	min. r	m	min. n	min. r
2	1	2	4	1	4
3	2	3	5	2	5
$4 \le m < 7$	3	4	$6 \le m < 9$	3	6
$7 \le m < 14$	4	5	$9 \le m < 16$	4	7
$14 \le m < 29$	5	6	$16 \le m < 31$	5	8
$29 \le m < 60$	6	7	$31 \le m < 62$	6	9
$60 \le m < 123$	7	8	$62 \le m < 125$	7	10
$123 \le m < 250$	8	9	$125 \le m < 252$	8	11
$250 \le m < 505$	9	10	$252 \le m < 507$	9	12

and 4, respectively. Then $\widetilde{\mathbf{H}}^{\mathrm{T}}$ is

$$\widetilde{\mathbf{H}}^{\mathrm{T}} = \begin{bmatrix} \mathbf{B} \\ \mathbf{I}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{E}_1 \ \mathbf{R}' \\ \mathbf{I}_4 \end{bmatrix} = \begin{bmatrix} 1 & \vdots & r'_{00} & r'_{01} & r'_{02} \\ 0 & \vdots & r'_{10} & r'_{11} & r'_{12} \\ 0 & \vdots & r'_{20} & r'_{21} & r'_{22} \\ 1 & \vdots & r'_{30} & r'_{31} & r'_{32} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & & 0 & 0 & 0 \\ 0 & & 1 & 0 & 0 \\ 0 & & 0 & 1 & 0 \\ 0 & & 0 & 0 & 1 \end{bmatrix}$$

where r'_{ij}, $0 \le i \le 3$ and $0 \le j \le 2$, are entries of $\mathbf{R}'$. The values of these entries are to be determined such that we have a single error correcting code. Among the total $N = 7 \cdot 6 \cdot 4 \cdot 3 = 504$ choices of different parity check matrices, one is

$$\widetilde{\mathbf{H}}^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{28}$$

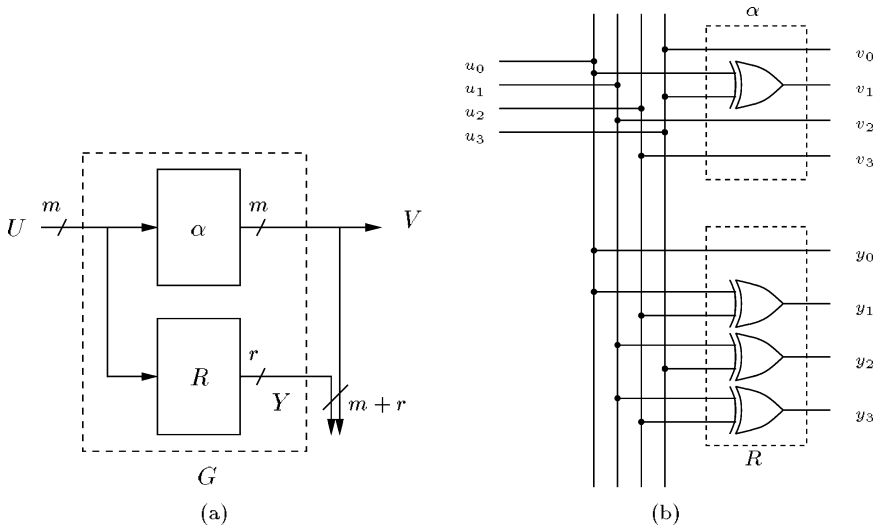


Fig. 2. (a) The block diagram of the code generator. (b) The circuit of Example 1

From $\widetilde{\mathbf{H}}^{\mathrm{T}}$, one can then write

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Using this $\mathbf{G}$, a circuit to generate the codeword is shown in Figure 2.

Comments:

—The solution shown in (28) is selected so that the total number of ones in $\mathbf{R}'$ is minimal resulting in the least number of gates in the architecture. Among other possible selections, one may select an $\mathbf{R}'$ such that the maximum number of ones in any column of $\mathbf{R}'$ is minimal to minimize the time delay. This is further discussed in Section 3.4.

—As an alternative to the proposed error correction scheme, one may use the single-bit error correcting Hamming code [Rao and Fujiwara 1989]. In this case, the R module of Figure 2(a) should be replaced with a module, say R_{H}, which generates parity bits of the Hamming code. In this case the inputs of the R_{H} module should come from the outputs of the α module. Also, the α module has to be fault free because a single stuck-at fault produces more than one error at the output of the R_{H} module. Although the number of parity bits of the Hamming code-based scheme is less than that of the scheme presented here,[1] the former requires more XOR gates. For example, when $m = 4$, the realization of the R_{H} module requires three parity bits and nine XOR gates.

[1]In the case of trinomials, the difference of the number of parity bits of two schemes is at most 1. This is because the minimum number of parity bits for the Hamming code, that is, r_H, satisfies the

This gate count is more than that of the scheme presented here (refer to Example 1).

Another advantage of the proposed scheme is that one can apply the following selection criteria to achieve certain VLSI design goals.

3.4 Selection Criteria for $\mathbf{R}$

As stated in Lemma 3.1, there are N possible choices for the parity matrix $\mathbf{R}$. Let us denote such possibilities as $\mathbf{R}'_j,\ 1 \leq j \leq N$, where N can be obtained from (21). For the purpose of implementation, one particular parity matrix can be better than the others. Below we list a number of rules for obtaining $\mathbf{R}'$ (and hence $\mathbf{R}$) for achieving optimal space, delay and power designs.

Rule 1 (*Space Optimization*). Let $S_j,\ 1 \leq j \leq N$, be the number of 1's in $\mathbf{R}'_j$. Then choose $\mathbf{R}'_k$ such that S_k is minimum, that is, $S_k \leq S_j\ \forall j \neq k$.

Rule 2 (*Delay Minimization*). Let $t_j,\ 1 \leq j \leq N$, be the maximum number of 1's in a column of $\mathbf{R}'_j$. Then, choose $\mathbf{R}'_k$ such that t_k is minimum, that is, $t_k \leq t_j\ \forall j \neq k$.

Rule 3 (*Path Equalization*). Let $g_j,\ 1 \leq j \leq N$, be the maximum difference in the number of 1's in columns of $\mathbf{R}'_j$. Then, choose $\mathbf{R}'_k$ such that g_k is minimum, that is, $g_k \leq g_j\ \forall j \neq k$.

Rule 1 yields the minimum number of XOR gates to generate the parity bits and more details about it are given in Section 5.1. Rule 2 results in the circuit for generating the parity bits with the lowest propagation delay caused by XOR gates. Rule 3 allows us to minimize power by path equalization [Benini et al. 2001]. This technique ensures that signal propagation from inputs to outputs of the parity generator block follows paths of similar length.

4. FAULT-TOLERANT SUM AND PASS-THRU MODULES

The sum module of Figure 1(a) is a finite field adder, which produces sum of the two elements of $GF(2^m)$ as its output. Let $A = (a_0,\ a_1,\ \ldots\ ,\ a_{m-1})$ and $B = (b_0,\ b_1,\ \ldots\ ,\ b_{m-1})$ be two inputs to this module as shown in Figure 1(b). Then the output is $C = A + B = (c_0,\ c_1, \ldots,\ c_{m-1})$, where $c_i = a_i + b_i$ for $0 \leq i \leq m-1$. An architecture of this module using m two-input XOR gates is shown in Figure 1(b).

In order to correct a single error in the sum module, a number of parity bits are needed. Suppose r parity bits are added to each of the two original m-bit inputs to form a codeword of $m+r$ bits so that a single error can be corrected. The procedure for producing r parity bits is similar to the one described earlier for the α module. If we extend the size of the sum module from m to $m+r$ and apply two $(m+r)$-bit inputs as codewords, then the output of the sum module will be a codeword as shown in Figure 3(a). In this figure, P_A, P_B, and P_C denote the parity bits associated with A, B, and C, respectively. Each of which has a length of r bits. The vector associated with P_A is calculated as $\underline{p_A} = \underline{a} \cdot \mathbf{R}$ and

inequality $2^{r_H} \geq r_H + m + 1$, which implies $2^{r_H} > r_H + m - 2$. When the latter is compared with (23), we have $r_H = n = r - 1$.

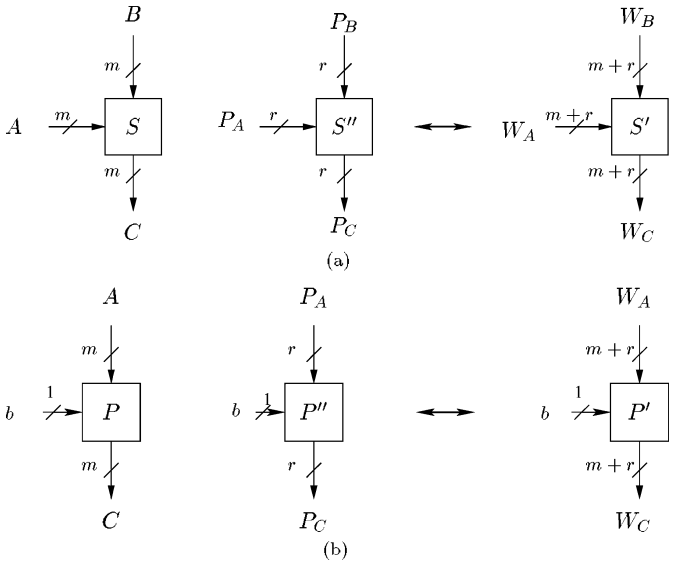


Fig. 3. (a) Single error correcting sum module. (b) Single error correcting pass-thru module.

similarly for P_B. By appending P_A to A, the codeword of A, which is denoted W_A is obtained. Similarly, W_B is obtained from P_B and B. It is worth to emphasis here that single fault in the above new sum module (which is shown by S') changes only a single bit of the output of the circuit. Thus, the error due to a single fault in the new sum module can be corrected using the same method described with the α module.

The pass-thru module of Figure 1(c) 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. Thus, the output of this module C is zero when $b = 0$ and A when $b = 1$. For correcting a single error due to a single stuck-at fault in this module, the codeword W_A of A is fed to the new pass-thru module that has $m + r$ two-input AND gates as shown in Figure 3(b). The single bit b is connected to the second input of each of these AND gates. So, the output of this new pass-thru module W_C is zero if $b = 0$ and W_A if $b = 1$. Since zero and W_A are codewords, W_C is a codeword, which can correct single errors. Since a single stuck fault in the new pass-thru module causes a single-bit error in the output of this module, this error can be corrected by again applying the same method discussed earlier in connection with the α module.

5. FAULT-TOLERANT POLYNOMIAL BASIS MULTIPLIER

The discussions of the previous sections dealt with the correction of errors in individual modules. Here, we combine the results of the previous sections for correcting errors in the entire multiplier.

5.1 Bit-Parallel Polynomial Basis Multiplier

In order to find a bit-parallel multiplier with a single error correction capability, all the α modules in Figure 1(a) can be replaced by the G module of Figure 2(a).

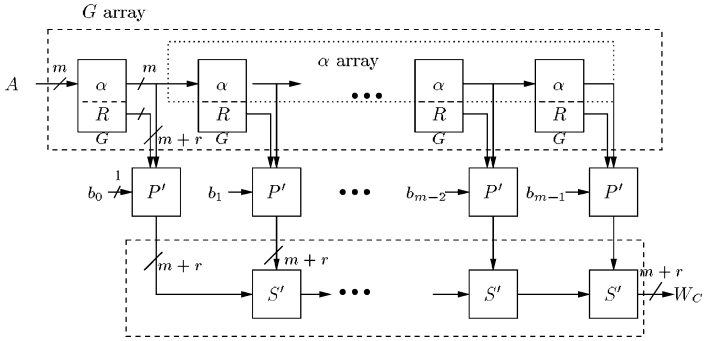


Fig. 4. The architecture of the bit-parallel polynomial basis multiplier with correction capability.

This is shown in Figure 4. The A input is also multiplied with the $\mathbf{G}$ matrix to obtain a codeword. The new sum and pass-thru modules of Figure 4 are from Figures 3(a) and (b), respectively.

Let $\underline{w_c}$ be the row vector associated with W_C in Figure 4. Then we have

$$\begin{aligned} \underline{w_c} &= b_0\underline{a}\mathbf{G} + b_1\underline{a}\mathbf{M}\mathbf{G} + \cdots + b_{m-1}\underline{a}\mathbf{M}^{m-1}\mathbf{G} \\ &= (b_0\underline{a} + b_1\underline{a}\mathbf{M} + \cdots + b_{m-1}\underline{a}\mathbf{M}^{m-1})\mathbf{G}. \end{aligned} \tag{29}$$

Using (8) and (4), we have

$$\underline{c} = b_0\underline{a} + b_1\underline{a}\mathbf{M} + \cdots + b_{m-1}\underline{a}\mathbf{M}^{m-1}. \tag{30}$$

Combining (29) and (30), yields

$$\underline{w_c} = \underline{c} \cdot \mathbf{G}. \tag{31}$$

It is easily seen that $\underline{w_c}$ is a codeword and so one error in any of its coordinates can be corrected. One can also see that a single stuck-at fault in any gate of the new sum, pass-thru (S' and P') modules, all $\mathbf{R}$ modules and the rightmost α module of Figure 4, causes only one-bit error at the output, which can be corrected. To *only* detect the errors due to such faults in the polynomial basis multiplier, one can see Reyhani-Masoleh and Hasan [2002].

It is worth mentioning here that not only a single fault in the abovementioned modules of Figure 4 produces a single error at the output but also any number of faults on a single coordinate (say jth coordinate, $0 \le j \le m-1$) affects only c_j at the output and hence this incorrect c_j can be corrected using the single error correcting method proposed here.

In order to calculate the overhead cost of the multiplier shown in Figure 4, one needs to consider overheads in each new module. Let S be the number of 1's in the $\mathbf{R}'$ matrix. Recall that $r = n+\omega-2$ is the number of parity bits generated by matrix $\mathbf{R} = [\mathbf{E}_0 \vdots \mathbf{R}']$. Thus, each fault-tolerent α module needs $S-n$ extra XOR gates. Since we have r parity bits, the number of extra gates related to each fault-tolerent pass-thru and sum modules are r AND gates and r XOR gates, respectively. Therefore, the overhead of the fault-tolerent PB multiplier would be mr AND gates and $m(S-n+r) = m(S+\omega-2)$ XOR gates (that is, $m(S+r+\omega-2)$ gates in total). Note that, the three rules stated in Section 3.4 to select $\mathbf{R}'$ yield fault-tolerent multipliers with the minimum number of

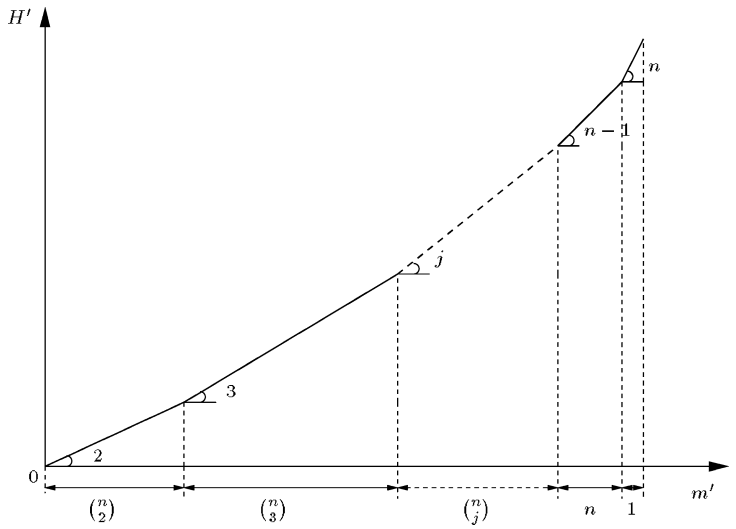


Fig. 5. Plot of H' versus m'.

XOR gates, time delay and power consumption. However, in order to minimize the total number of gates in the proposed fault-tolerent multiplier, one should obtain $\mathbf{R}'$ such that $(S + r + \omega - 2)$ is minimum.

It is noted that the overhead of the TMR multiplier based on Figure 1(a) is at least twice the cost of the basic multiplier, that is, $2m(2m+\omega-3)$ gates in total. Therefore, for space efficiency, we have to choose $\mathbf{R}'$ such that the overhead of the proposed multiplier in Figure 4 is less than that of the TMR, that is, $S < 4m - n - 2$. Below, we present a lemma for minimum on S for a given n that satisfies Theorem 3.2.

LEMMA 5.1. *For a given n that satisfies Theorem* 3.2*, a minimum value of S is*

$$\min\{S\} = \begin{cases} 2 + H' & \text{for } \omega = 3 \\ \omega - 2 + H' & \text{for } 3 < \omega \le m \\ m - 1 & \text{for } \omega = m + 1 \end{cases}$$

where H' is a piecewise linear function of $m' = m - \omega + 1$ whose breakpoints are functions of n as

$$H' = \begin{cases} 2m' & \text{if } m' \le \binom{n}{2} \\ jm' - \sum_{i=2}^{j-1}(j-2)\binom{n}{i} & \text{if } 0 < m' - \sum_{i=2}^{j-1}\binom{n}{i} \le \binom{n}{j},\ 3 \le j \le n. \end{cases} \tag{32}$$

A proof of this lemma is given in Appendix B for convenience. Figure 5 shows a parametric plot of H' versus m'.

5.2 Correction Procedure

For the sake of simplicity, we use $\hat{W}$ and corresponding vector $\underline{\hat{w}}$ instead of $\hat{W}_C$ and $\underline{\hat{w}_c}$, respectively. A block diagram for correcting an error in the fault-tolerent polynomial basis multiplier is shown in Figure 6(a), where $\hat{W}$ denotes

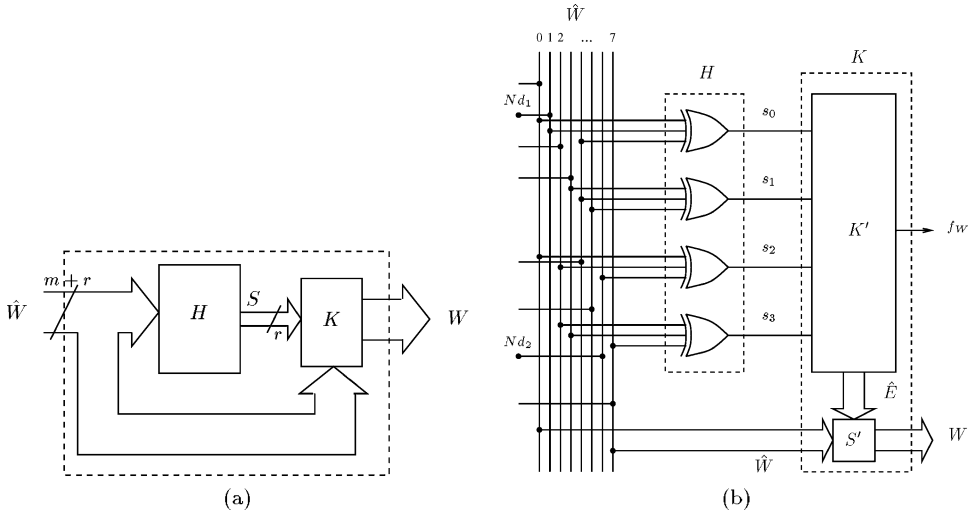


Fig. 6. (a) Correction module. (b) The complete circuit of Example 2.

the erroneous multiplier output. In the absence of any fault, $\underline{\hat{w}}$ is the same as $\underline{w}$ of (31). So, the $\underline{w}$ and $\underline{\hat{w}}$ vectors are the outputs of the fault-free and faulty polynomial basis multiplier (Figure 4), respectively. We assume that the error due to a single fault in the original circuit changes $\underline{w}$ to $\underline{\hat{w}}$ based on the following equation:

$$\underline{w} = \underline{\hat{w}} + \underline{\hat{e}} \tag{33}$$

where $\underline{\hat{e}}$ is the error row vector related to the output row vector $\underline{\hat{w}}$. The length of $\underline{\hat{e}}$ is $m + r$ and its entry is one where the error has occurred. To correct $\underline{\hat{w}}$ to $\underline{w}$, we can apply the following procedure commonly used in the decoding of a linear block code [see, e.g., Reed and Chen 1999].

The H module computes the syndrome of the received vector as $\underline{s} = \underline{\hat{w}} \cdot \mathbf{H}^{\mathrm{T}}$. Using (33) and $\underline{w} \cdot \mathbf{H}^{\mathrm{T}} = 0$, we have $\underline{s} = \underline{\hat{e}} \cdot \mathbf{H}^{\mathrm{T}}$.

If there is no error in $\hat{w}$, $\underline{s}$ is 0. If one error has occurred, $\underline{\hat{e}}$ has only a 1 in the position of the error. Then, $\underline{s}$ is equal to one column of $\mathbf{H}$ whose position determines the position of the nonzero entry in $\underline{\hat{e}}$. This is realized by the K module in Figure 6(b). In the first part of the K module, $\underline{\hat{e}}$ is obtained by comparing $\underline{s}$ with columns of $\mathbf{H}$ and in the next part, $\underline{\hat{w}}$ is corrected to $\underline{w}$ using (33), which is implemented by $(m + r)$ two-input XOR gates.

The following example shows how the above procedure works in response to certain faults in the circuit given in Figure 4.

Example 2. The correction module including the H and K modules of the polynomial basis multiplier used in Example 1 is shown in Figure 6(b).

Using $\mathbf{H}$ from Example 1, we can write

$\underline{s} = \underline{\hat{w}} \cdot \mathbf{H}^{\mathrm{T}} = [\hat{w}_0 + \hat{w}_1 + \hat{w}_4,\ \hat{w}_3 + \hat{w}_4 + \hat{w}_5,\ \hat{w}_0 + \hat{w}_2 + \hat{w}_6,\ \hat{w}_2 + \hat{w}_3 + \hat{w}_7]$.

The K module consists of a combinational logic K' and a sum module S'. The combinational logic K', whose truth table is shown in Table II, compares the vector $\underline{s}$ with the columns of $\mathbf{H}$ to produce $\underline{\hat{e}}$. Then, using (33), $\underline{\hat{w}}$ is corrected

Table II. Truth Table of Combinational Logic K' of Figure 6

s_0	s_1	s_2	s_3	$\hat{e}_0$	$\hat{e}_1$	$\hat{e}_2$	$\hat{e}_3$	$\hat{e}_4$	$\hat{e}_5$	$\hat{e}_6$	$\hat{e}_7$	f_W
1	0	1	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0
Otherwise				x	x	x	x	x	x	x	x	1

x = do not care.

to $\underline{w}$. This is accomplished by the sum module S' in K. If $\underline{s}$ does not match any column of $\mathbf{H}$, then a flag, f_W, is activated to indicate that more than one error has occurred. In this case, the output $\underline{w}$ is ignored.

Now, we want to show how this architecture can correct the error due to multiple faults in either lines of Nd_1 or Nd_2 in Figure 6(b). Let $\underline{w} = [1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1]$ be the output vector of the multiplier of Figure 4 for Example 1. Suppose the bit 1 of the output, that is, node Nd_1, is changed to 0. Then, $\underline{\hat{w}} = [1,\ \underline{0},\ 1,\ 0,\ 0,\ 0,\ 0,\ 1]$ and $\underline{s} = [1,\ 0,\ 0,\ 0]$. Using Table II, $\underline{\hat{e}} = [0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]$ and the second error bit in $\underline{\hat{w}}$ is corrected to $\underline{w} = \underline{\hat{w}} + \underline{\hat{e}} = [1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1]$.

This architecture also corrects an error in the parity bits of the output. Let a s-a-1 be the fault at node Nd_2, or any other multiple faults that result in the same error, then $\underline{\hat{w}} = [1, 1, 1, 0, 0, 0, \underline{1}, 1]$, $\underline{s} = [0, 0, 1, 0]$, $\underline{\hat{e}} = [0, 0, 0, 0, 0, 0, 1, 0]$, and $\underline{w} = [1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1]$, which is the correct codeword.

Finally, assume that both of the above errors have occurred. The vectors would be as follows:

$\underline{\hat{w}} = [1,\ \underline{0},\ 1,\ 0,\ 0,\ 0,\ \underline{1},\ 1]$, $\underline{s} = [1,\ 0,\ 1,\ 0]$, $\underline{\hat{e}} = [1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]$, and $\underline{w} = [\underline{0},\ \underline{0},\ 1,\ 0,\ 0,\ 0,\ \underline{1},\ 1]$.

In this case, the output is wrong, that is, the circuit only corrects single errors and detects some of the double errors.

5.3 Extension to Bit-Serial Multiplier

Using the architecture of Figure 4, a bit-serial polynomial basis multiplier with a single error correction capability is shown in Figure 7. In this figure, Z is an $m + r$ bit register, which is initialized by the codeword of A where the top m bits of this register are the coordinates of A and other r bits are the parity bits. The correction module is the same as shown in Figure 6(a) and is assumed to be fault free. It can correct any single error that occurs in its input during a clock cycle. Thus, in the fault-free situation, the output of this block is the same as its input and one can easily find that after m clock cycles the top m bits of the correction module will contain the coordinates of finite field multiplication $C = AB$. Suppose a single stuck-at fault occurs in bit j of any module of Figure 7 (except in the correction module), then only the jth coordinate of Z, that is, z_j, may be erroneous and is corrected by the correction module. In the next clock

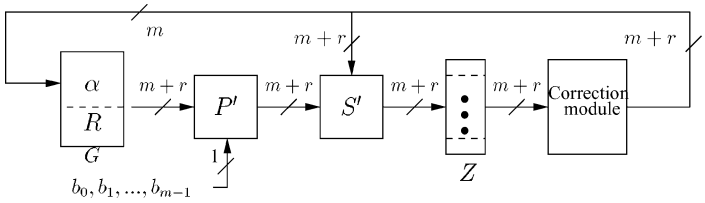


Fig. 7. The architecture of the bit-serial polynomial basis multiplier with the correction capability.

cycle, we may have an error in the jth coordinate and is corrected again by the correction module. In spite of having an error in the jth coordinate in each clock cycle, the output of correction module is error free. Thus, after the mth clock cycle the top m bits of the output of the correction module will contain the coordinates of C.

6. CONCLUDING REMARKS

Finite field arithmetic is extensively used in a number of cryptosystems. Faults in a hardware unit of such field arithmetic reduce yield factors and, more importantly, may make the cryptosystem vulnerable against side-channel attacks. In this article, we have considered fault-tolerent multiplication in the finite field $GF(2^m)$. To the best of our knowledge, no previous article has addressed this issue of correcting errors in the finite field multiplier. Here we have presented pertinent theoretical results for correcting single errors caused by one or more faults in the multiplier circuits.

For the error-correction scheme, we have given a lower bound on the number of parity bits needed and an exact expression for the total number of parity check matrices. For better hardware implementation, we have listed a set of rules for space optimization, logic gate delay minimization, and signal path equalization. It has been shown that by using a parity matrix of low Hamming weight, the overhead cost of the fault-tolerent multiplier can be lower than that of a conventional TMR based multiplier.

The results presented here are quite generic in the sense that they can be applied to any binary polynomials used for defining the field. We have presented fault-tolerent architectures for both bit-parallel and bit-serial field multipliers. By combining the proposed fault-tolerent bit-serial and bit-parallel multipliers, one can develop fault-tolerent hybrid multipliers over composite finite fields.

APPENDIX

A. PROOF OF LEMMA 3.1

For having single error correction capability, no two columns of $\mathbf{H}$ (or $\widetilde{\mathbf{H}}$) would be the same. For the purpose of this proof, it appears easier to deal with rows. So, instead of $\widetilde{\mathbf{H}}$, here we use $\widetilde{\mathbf{H}}^{\mathrm{T}}$, that is,

$$\widetilde{\mathbf{H}}^{\mathrm{T}} = \begin{bmatrix} \mathbf{B} \\ \mathbf{I_r} \end{bmatrix}. \tag{A.1}$$

Since the $(\omega-2)$ columns of $\mathbf{E}_1$ are known, the rows of $\mathbf{B}$ can be categorized into the following three groups:

(1) One row whose first $(\omega-2)$ entries are all 1's.
(2) $(\omega-2)$ rows each with only one 1 in the first $(\omega-2)$ entries. The position of this 1 is ρ_j, $1 \le j \le \omega-2$.
(3) $m-(\omega-1)$ rows with zeros in the first $(\omega-2)$ entries.

Let $\underline{b}_1$ be the row vector associated with group 1, that is,

$$\underline{b}_1 = \Big[\underbrace{1\ 1\ \ldots\ 1}_{\omega-2}\ \underline{r}'_1 \Big]. \tag{A.2}$$

Let $\mathbf{B}_2$ and $\mathbf{B}_3$ denote two matrices obtained by using the rows of groups 2 and 3, respectively. Thus

$$\mathbf{B}_2 = \big[\, \mathbf{I}_{\omega-2}\ \ \mathbf{R}'_2 \,\big] \tag{A.3}$$

$$\mathbf{B}_3 = \big[\, 0_{m-(\omega-1)\times(\omega-2)}\ \ \mathbf{R}'_3 \,\big]. \tag{A.4}$$

Since $r = n + (\omega-2)$, we also have

$$\mathbf{I_r} = \begin{bmatrix} \mathbf{I}_{\omega-2} & 0_{(\omega-2)\times n} \\ 0_{n\times(\omega-2)} & \mathbf{I_n} \end{bmatrix}. \tag{A.5}$$

Case 1 $(3 < \omega \le m)$

Group 1: The only row of $\widetilde{\mathbf{H}}^{\mathrm{T}}$ that has all 1's in the first $(\omega-2)$ entries is shown in (A.2). Thus the n entries of $\underline{r}'_1$ can be any combination of 0 and 1. Thus, the number of different selections for this group is

$$N_1 = 2^n. \tag{A.6}$$

Group 2: Referring to (A.3) and (A.4), it is clear that $(\omega-2)\times n$ matrix $\mathbf{R}'_2$ cannot have a row with all entries being 0. Thus the possible number choices for any row is 2^n-1. Since selection of all $(\omega-2)$ rows of $\mathbf{R}'_2$ are independent from each other, the number of different selections for this group is

$$N_2 = (2^n-1)^{\omega-2}. \tag{A.7}$$

Group 3: From (A.4) and (A.5), one can see that $\mathbf{R}'_3$ should have at least two 1's in each row. We have to multiply the number of selections of $m-(\omega-1)$ rows to get the total number of selection for $\mathbf{R}'_3$. Let us start from row 0. Since row 0 cannot be any rows of $\mathbf{I_n}$ and cannot have all zero entries, there are $2^n-(n+1)$ possible selections for this row. For row 1, the additional constrain is that it cannot be row 0. So, the number of selections for this row is $2^n-(n+2)$. In general, the number of selections for row i, $0 \le i \le m-\omega$, is $2^n-(n+1+i)$. By multiplying these numbers and changing $i+1$ to j, the number of selections for this group is calculated as

$$N_3 = \prod_{j=1}^{m-\omega+1} (2^n-(n+j)). \tag{A.8}$$

Therefore, the total number of different parity check matrices $\mathbf{H}$ is obtained by multiplying all possible selections of the three groups, that is,

$$N = N_1 \cdot N_2 \cdot N_3$$

which completes the proof of the first case of (21).

Case 2 ($\omega = 3$):

This is the case of trinomials. Without any loss of generality, assume that $F(z) = z^m + z + 1$. Then, group 2 has only one row whose first entry is a single 1 similar to group 1 (refer to Case 1). As a result, there are two rows with a single one in the first column of $\mathbf{B}$. Then the number of selections of these two rows is found by extracting the selections of zero entries which are different, that is,

$$N_{1,2} = (2^n - 1)\,(2^n - 2). \tag{A.9}$$

Using (A.9) and (A.8),

$$N = N_{1,2} \cdot N_3\ ,$$

and the proof is done for $\omega = 3$.

Case 3 ($\omega = m + 1$):

For $\omega = m + 1$ (that is, all-one-polynomial), we only have the first and second groups of Case 1. So,

$$N = N_1 \cdot N_2\ ,$$

and the proof is complete. □

B. PROOF OF LEMMA 5.1

Let us recall the transposition of the parity check matrix which is

$$\widetilde{\mathbf{H}}^{\mathrm{T}} = \begin{bmatrix} \mathbf{E}_1 & \mathbf{R}' \\ & \mathbf{I_r} \end{bmatrix}. \tag{B.1}$$

We should obtain the $m \times n$ matrix $\mathbf{R}'$ such that no two rows of (B.1) are the same and the number of 1's in $\mathbf{R}'$, that is, S, is minimum. Based on the three different rows of $\mathbf{E}_1$ in (20) (see Appendix A for more information), we can fill entries of $\mathbf{R}'$ as follows.

For $\omega > 3$:

(1) All entries of the last row of $\mathbf{R}'$ should be zero, because all entries of the last row of $\mathbf{E}_1$ are 1s and there is no such a row in $\widetilde{\mathbf{H}}^{\mathrm{T}}$ of (B.1).
(2) In $\mathbf{E}_1$, $\omega - 2$ rows have only one '1' each. The corresponding $\omega - 2$ rows in $\mathbf{R}'$ also need to have only one '1' each. This is because these rows should be different from $\omega - 2$ rows of $\mathbf{I_r}$, which have only one '1' in each row.
(3) The remaining $m' = m - \omega + 1$ rows of $\mathbf{R}'$ should have at least two 1s each. This is to ensure that the corresponding rows of $\widetilde{\mathbf{H}}^{\mathrm{T}}$ are different from other rows. To minimize S, first we fill the entries of these rows with only two 1s.

The maximum number of possibilities of doing so is $\binom{n}{2}$. If $m' > \binom{n}{2}$, then we should fill the remaining $m' - \binom{n}{2}$ rows of $\mathbf{R}'$ with three 1s. If there still exists any row, that is, if $m' - \binom{n}{2} > \binom{n}{3}$, we should filling the remaining rows with four 1s, and so on. Based on the above discussions, one can obtain the total number of 1s in the mentioned m' rows as follows:

$$H' = \begin{cases} 2m' & \text{if } m' \le \binom{n}{2} \\ 3m' - \binom{n}{2} & \text{if } 0 < m' - \binom{n}{2} \le \binom{n}{3} \\ 4m' - 2\binom{n}{2} - \binom{n}{3} & \text{if } 0 < m' - \binom{n}{2} - \binom{n}{3} \le \binom{n}{4} \\ 5m' - 3\binom{n}{2} - 2\binom{n}{3} - \binom{n}{4} & \text{if } 0 < m' - \binom{n}{2} - \binom{n}{3} - \binom{n}{4} \le \binom{n}{5} \\ \vdots & \vdots \end{cases}$$

which simplifies to (32).

Thus, by adding 1s in the above three parts of $\mathbf{R}'$, one can obtain $\min\{S\} = \omega - 2 + H'$.

For $\omega = 3$: The ρ_1th row and the last row of $\mathbf{R}'$ should have only one 1 in two different entries. Thus, one can obtain the minimum on S as $2 + H'$ (see Example 1 for more details).

For $\omega = m + 1$: For an all-one polynomial, $\omega = m + 1$; thus in Theorem 3.2, the lowest value of n is 1. For an all-one polynomial, we also have, $\rho_j = j,\ 1 \le j \le m - 1$.

For $n = 1$, two possibilities of the $m \times m$ parity matrix $[\mathbf{E}_1 \vdots \mathbf{R}']$ can be obtained as

$$[\mathbf{E}_1 \vdots \mathbf{R}'] = \begin{bmatrix} 1 & 0 & \dots & 0 & 1 \\ 0 & 1 & \dots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & r' \end{bmatrix}, \tag{B.2}$$

where r' is either 0 or 1. Thus, for $r' = 0$, there are $m - 1$ 1's in $\mathbf{R}'$. It is noted that if $n > 1$ is chosen, the minimum number of 1s in $\mathbf{R}'$ does not change.

REFERENCES

BENINI, L., MICHELI, G. D., AND MACII, E. 2001. Designing low-power circuits: Practical recipes. *IEEE Circ. Syst. Mag. 1*, 1, 6–25.

FENN, S., GOSSEL, M., BENAISSA, M., AND TAYLOR, D. 1998. Online error detection for bit-serial multipliers in $GF(2^m)$. *J. Electron. Test.: Theory Applic. 13*, 29–40.

HALBUTOGULLARI, A. AND KOC, C. K. 2000. Mastrovito multiplier for general irreducible polynomials. *IEEE Transactions on Computers 49*, 5 (May), 503–518.

JOHNSON, B. W. 1989. *Design and Analysis of Fault-Tolerant Digital Systems*. Addison-Wesley Publishing Company, Reading, MA.

LALA, P. K. 1985. *Fault Tolerant and Fault Testable Hardware Design*. Prentice Hall, Englewood Cliffs, NJ.

LIDL, R. AND NIEDERREITER, H. 1994. *Introduction to Finite Fields and Their Applications*. Cambridge University Press.

LIN, S. AND COSTELLO, D. J. 1983. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, NJ.

MASTROVITO, E. D. 1991. VLSI Achitectures for Computation in Galois Fields. Ph.D. thesis, Linkoping University, Linkoping Sweden.

MENEZES, A. J., BLAKE, I. F., GAO, X., MULLIN, R. C., VANSTONE, S. A., AND YAGHOOBIAN, T. 1993. *Applications of Finite Fields*. Kluwer Academic Publishers, Boston, MA.

PRADHAN, D. K. 1996. *Fault Tolerant Computer System Design*. Prentice Hall, Englewood Cliffs, NJ.

RAO, T. R. N. AND FUJIWARA, E. 1989. *Error Control Coding for Computer Systems*. Prentice Hall, Englewood Cliffs, NJ.

REED, I. S. AND CHEN, X. 1999. *Error-Control Coding for Data Networks*. Kluwer Academic Publishers, Boston, MA.

REYHANI-MASOLEH, A. AND HASAN, M. A. 2002. Error detection in polynomial basis multipliers over binary extension fields. In *Cryptographic Hardware and Embedded Systems—CHES 2002*, Lecture Notes in Computer Science, Vol. 2528, Springer Verlag, Berlin, Germany, 515–528.

VANSTONE, S. A. AND VAN OORSCHOT, P. C. 1989. *An Introduction to Error Correcting Codes with Applications*. Kluwer Academic Publishers, Boston, MA.

ZHANG, T. AND PARHI, K. K. 2001. Systematic design of original and modified mastrovito multipliers for general irreducible polynomials. *IEEE Trans. Comput. 50*, 7 (July), 734–748.

Received February 2003; revised May 2003, June 2003, and July 2003; accepted July 2003