Reliable Concurrent Error Detection Architectures for Extended Euclidean-Based Division Over GF(2^m)

Mehran Mozaffari-Kermani, *Member, IEEE*, Reza Azarderakhsh, Chiou-Yng Lee, *Senior Member, IEEE*, and Siavash Bayat-Sarmadi, *Member, IEEE*

***Abstract*— The extended Euclidean algorithm (EEA) is an important scheme for performing the division operation in finite fields. Many sensitive and security-constrained applications such as those using the elliptic curve cryptography for establishing key agreement schemes, augmented encryption approaches, and digital signature algorithms utilize this operation in their structures. Although much study is performed to realize the EEA in hardware efficiently, research on its reliable implementations needs to be done to achieve fault-immune reliable structures. In this regard, this paper presents a new concurrent error detection (CED) scheme to provide reliability for the aforementioned sensitive and constrained applications. Our proposed CED architecture is a step forward toward more reliable architectures for the EEA algorithm architectures. Through simulations and based on the number of parity bits used, the error detection capability of our CED architecture is derived to be 100% for single-bit errors and close to 99% for the experimented multiple-bit errors. In addition, we present the performance degradations of the proposed approach, leading to low-cost and reliable EEA architectures. The proposed reliable architectures are also suitable for constrained and fault-sensitive embedded applications utilizing the EEA hardware implementations.**

***Index Terms*— Efficient fault diagnosis, error coverage (EC), extended Euclidean algorithm (EEA), reliable and constrained embedded systems.**

I. Introduction

CURRENTLY, many sensitive and security-constrained applications and embedded systems such as those based on Radio Frequency IDentification technology or Near-Field Communication require high-performance, low-power, and energy-aware architectures for the cryptographic solutions providing them with the level of security required. Not only do these cryptographic hardware and embedded systems security mechanisms need to be efficient, but they also need to be able to thwart the implementation attacks taking advantage of the side-channel information leaked.

Various research works are recently focused on achieving high-performance finite field arithmetic implementations because of their use in many applications, e.g., the elliptic curve cryptography (ECC) (developed independently by Miller and Koblitz). One of the paramount advantages of the ECC over conventional public-key schemes is that much smaller key sizes are required to achieve the same security level [1], [2]. Several ECC implementations were reported in the literature at the hardware level, see [3]–[6]. The ECC arithmetic operations include multiplications and divisions over GF(2^m) and GF(p) [1], which are highly time-consuming. Thus, it is desirable to develop efficient division architectures for GF(2^m).

In finite field GF(2^m), a number of schemes are mainly applied for computing the division operation, e.g., the Fermat's little theorem [7], the discrete-time Wiener-Hopf equation (DTWHE), and the extended Euclidean algorithm (EEA) [8]–[12]. For Fermat's little theorem approaches, the division A/B can be represented by repeating multiply-square algorithms such that $AB^{-1} = AB^{2^m-2} = A(B(B\cdots(B(B)^2)^2\cdots)^2)^2$. This method requires $m-1$ times squarings and m times multiplications for implementing systolic architecture [7]. The second method, i.e., DTWHE, finds a multiplicative inversion in GF(2^m) by solving a system of $2m-1$ linear equations with $2m-1$ unknowns over GF(2) through Gaussian elimination. The last method, i.e., the EEA, uses the fact that GCD($G(x), B$) $= 1$, where B is a nonzero element in GF(2^m) and $G(x)$ is an irreducible polynomial to generate the field. The EEA finds R and S to satisfy $RG + SB = 1$.

Although a number of very-large scale integration (VLSI) implementations were presented for the EEA scheme, see [8]–[11], they do not guarantee that in presence of natural or malicious faults, the architectures are reliable and fault-immune. In fact, both natural faults caused by defects in the VLSI implementations and also fault attacks (a powerful subset of side-channel techniques that is used to break cryptographic schemes) can compromise the reliability of the hardware implementations of the EEA. Boneh *et al.* [13] first introduced how to recover secret keys of the Rivest-Shamir-Adleman and the discrete-logarithm-based cryptosystems. This idea is to inject faults during computations and to use the erroneous outputs to deduce information on the secret

Manuscript received December 9, 2012; revised March 26, 2013; accepted April 21, 2013. Date of publication June 14, 2013; date of current version April 22, 2014.

M. Mozaffari-Kermani is with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: mozaffari@ieee.org).

R. Azarderakhsh is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: azarderakhsh@gmail.com).

C.-Y. Lee is with the Department of Computer Information and Network Engineering, Lunghwa University of Science and Technology, Taoyuan 33306, Taiwan (e-mail: pp010@mail.lhu.edu.tw).

S. Bayat-Sarmadi is with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (e-mail: bayatsr@gmail.com).

Digital Object Identifier 10.1109/TVLSI.2013.2260570

key stored in the secure component. In addition, in [14], differential fault analysis attacks are investigated for the ECC. We also note that in other cryptographic algorithms, natural and malicious faults can undermine the reliability of the implementations, see [15]–[17], for the ones on the advanced encryption standard (AES).

For cryptographic architectures such as the AES, many research works are carried out to achieve reliable and fault-immune structures, see [18]–[24] (also, refer to [25] for reliable architectures for a lightweight block cipher). In addition, concerning the finite field arithmetic architectures (for some finite field multiplication architectures refer to [26]–[30]), various concurrent error detection (CED) multipliers for polynomial basis and normal basis of GF(2^m) are proposed using parity codes and recomputing using shifted operands schemes [31]–[34].

In this paper, we propose a new multiple parity scheme to realize the CED EEA division architectures. The proposed parity scheme is denoted as dual parity prediction, detecting the errors in the division results. Our simulation results for five different number of parity bits and different number of both stuck-at-zero and -one injections show close to 100% error coverage for the proposed fault diagnosis approaches while reaching acceptable hardware, time, and efficiency overheads. We note that almost all of the occurring errors caused by natural faults are detected using the proposed methods. Although the proposed approaches may not result in a complete solution to the problem of intentionally injected faults, the high error coverage achieved would most likely make these attacks more difficult. We also note that the architecture of the CED systolic array inversion presented in [35] uses a single parity prediction scheme for fault diagnosis that results in suboptimal error coverage for multiple fault injections. Our proposed architecture can reach the error coverage (EC) of close to 100% for multiple injected faults. In addition, our proposed architecture can be tailored based on the reliability objectives and the overhead to be tolerated, i.e., multiple parity prediction signatures can be potentially utilized. With the available resources and the reliability goals to achieve, one can utilize the proposed architectures to have more reliable hardware architectures of the EEA divisions.

The organization of this paper is as follows. In Section II, mathematical background related to the polynomial basis division algorithm and parity codes are presented. Section III presents the proposed parity prediction schemes. In Section IV, the proposed CED architectures are presented. Simulation-based experiments for deriving the error coverage are presented in Section V. In addition, in this section, the complexity analysis and the overhead benchmark are presented. Finally, we conclude in Section VI.

II. Mathematical Background

This section briefly introduces the basic concepts of polynomial basis division algorithm and single parity code.

A. Polynomial Basis Division Algorithm

Finite fields of order 2^m are denoted as binary fields (also referred to as characteristic-two finite fields). One way to construct GF(2^m) is to use a polynomial basis representation. The elements of GF(2^m) are the binary polynomials (polynomials whose coefficients are in the field $\{0, 1\}$) of degree at most $m-1$, e.g., $A = a_0 + a_1x + \cdots + a_{m-1}x^{m-1}$, with coefficients in GF(2). We note that an irreducible binary polynomial of degree m is also chosen and used for arithmetic operations. Addition of field elements is performed with coefficient arithmetic performed modulo 2. Multiplication of field elements is performed modulo the reduction polynomial.

Let A and B be two elements in GF(2^m), $G(x)$ be the irreducible polynomial to construct the finite field GF(2^m), and C be the result of the division $A/B \bmod G(x)$. In other words, let us have

$$\begin{aligned} A &= a_0 + a_1x + \cdots + a_{m-1}x^{m-1} \\ B &= b_0 + b_1x + \cdots + b_{m-1}x^{m-1} \\ C &= c_0 + c_1x + \cdots + c_{m-1}x^{m-1} \\ G(x) &= g_0 + g_1x + \cdots + g_{m-1}x^{m-1} + x^m \end{aligned}$$

where the coefficients of each polynomial is in GF(2). To compute the division $A/B \bmod G(x)$, we have the followings:

1) if R and S are both even, then GCD$(R, S) = x$GCD$(R/x, S/x)$;
2) if R is even and S is odd, then GCD(R, S) = GCD$(R/x, S)$;
3) if R and S are odd, then GCD(R, S)=GCD$(R + S/x, S)$.

Considering the above facts, let $R^{(i)}, S^{(i)}, U^{(i)}, V^{(i)}$ be the ith recursive computational results of four polynomials R, S, U, V, respectively

$$\begin{aligned} R^{(i)} &= r_{i,m}x^m + r_{i,m-1}x^{m-1} + \cdots + r_{i,1}x + r_{i,0} \\ S^{(i)} &= s_{i,m}x^m + s_{i,m-1}x^{m-1} + \cdots + s_{i,1}x + s_{i,0} \\ U^{(i)} &= u_{i,m-1}x^{m-1} + u_{i,m-2}x^{m-2} + \cdots + u_{i,1}x + u_{i,0} \\ V^{(i)} &= v_{i,m-1}x^{m-1} + v_{i,m-2}x^{m-2} + \cdots + v_{i,1}x + v_{i,0}. \end{aligned}$$

In the initial step, assume that $R = B, S = G(x), U = A$, $V = 0$, and $\beta = -1$. Then, the division algorithm is performed through Algorithm 1.

In this algorithm, $(U/x)_G$ refers to $(U/x) \bmod G(x)$. Observing the division in Algorithm 1, two pairs (R, S) and (U, V) in each recursive operation are controlled by two values β and r_0 to determine simple exchange and shift operations. Basically, each recursive operation in Algorithm 1 is satisfied by $s_{i,0} = 1$. Therefore, from Steps 4, 7, and 10, the polynomial $R^{(i)}$ can be obtained as follows:

$$\begin{aligned} R^{(i)} &= \frac{(R^{(i-1)} + r_0S^{(i-1)})}{x} \\ &= \frac{1}{x}[(r_{i-1,0} + r_{i-1,1}x + \cdots + r_{i-1,m}x^m) \\ &\quad + r_0(s_{i-1,0} + s_{i-1,1}x + \cdots + s_{i-1,m}x^m)] \\ &= (r_{i-1,1} + r_0s_{i-1,1}) + (r_{i-1,2} + r_0s_{i-1,2})x \\ &\quad + \cdots + (r_{i-1,m} + r_0s_{i-1,m})x^{(m-1)} \\ &= r_{i,0} + r_{i,1}x + \cdots + r_{i,m}x^m \end{aligned} \quad (1)$$

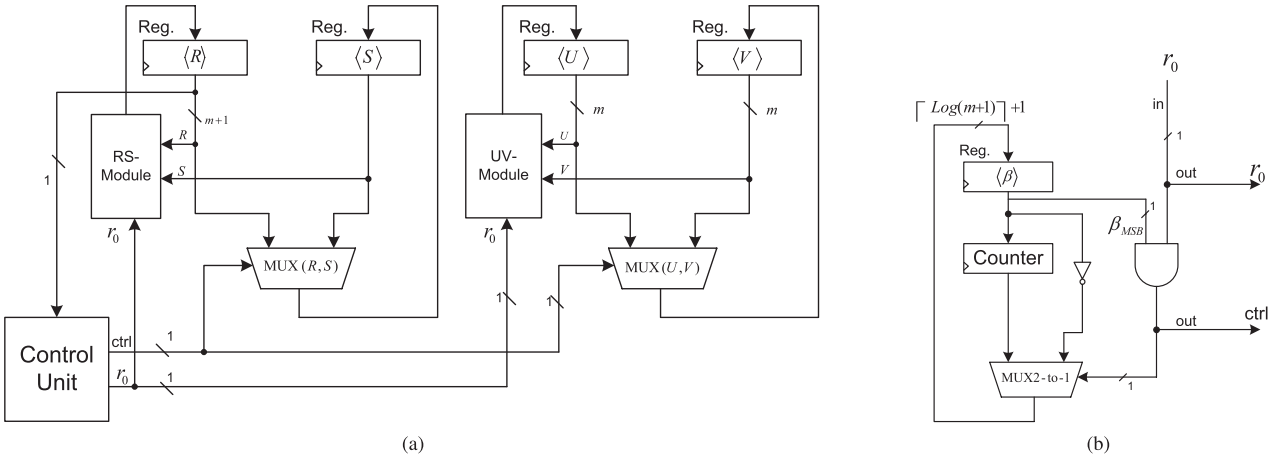

Fig. 1. Architectures for (a) sequential division and (b) its control unit.

Algorithm 1 Division algorithm presented in [9]

Inputs: $G(x), A, B$.
Output: $V = \frac{A}{B} \bmod G(x)$ (denoted also as $(\frac{A}{B})_G$).
Initialize: $R = B, S = G(x), U = A, V = 0, \beta = -1$.
1. for $k = 1$ to $2m - 1$ do
2. If $r_0 = 1$ then
3. If $\beta < 0$ then
4. $(R, S, U, V) \leftarrow (R + S, R, U + V, U)$.
5. β=-β.
6. else
7. $(R, S, U, V) \leftarrow (R + S, S, U + V, V)$.
8. end if
9. end if
10. $(R, U) \leftarrow (\frac{R}{x}, (\frac{U}{x})_G)$.
11. $\beta = \beta - 1$.
12. end for.

where

$$\begin{aligned} r_{i,m} &= 0 \\ r_{i,j} &= r_{i-1,j+1} + r_0 s_{i-1,j+1}, \quad \text{for } 0 \le j \le m-1 \\ r_0 &= r_{i-1,0}. \end{aligned}$$

As the field GF(2^m) is generated by the irreducible polynomial $G(x)$ with $g_0 = 1$, we have the following:

$$x^{-1} = g_1 + \cdots + g_{m-1}x^{m-2} + x^{m-1}.$$

Thus, the polynomial $U^{(i)}$ is computed as follows:

$$\begin{aligned} U^{(i)} &= \frac{(U^{(i-1)} + r_0 V^{(i-1)})}{x} \bmod G(x) \\ &= u_{i,0} + u_{i,1}x + \cdots + u_{i,m-1}x^{(m-1)} \end{aligned} \tag{2}$$

where

$$\begin{aligned} u_{i,j} &= u_{i-1,j+1} + r_0 v_{i-1,j+1} + g_{j+1}(u_{i-1,0} + r_0 v_{i-1,0}) \\ u_0 &= g_1(u_{i-1,0} + r_0 v_{i-1,0}). \end{aligned}$$

From two Steps 5 and 11, we can use the two's complement representation to count the value β, through $\lceil \log_2(m+1) \rceil + 1$ bits to control the GF(2^m) division. It is noted that ctrl $= \beta_{\text{MSB}} \cdot r_0$ (note that β_{MSB} is the most significant bit of β), the value β in each recursive division operation is given by the following relation:

$$\beta = \begin{cases} \beta - 1, & \text{if ctrl} = 0 \\ (-\beta) - 1 = (\overline{\beta} + 1) - 1 = \overline{\beta}, & \text{if ctrl} = 1. \end{cases} \tag{3}$$

Therefore, based on the computation of two polynomials S and V in two Steps 4 and 7 of Algorithm 1, we can obtain the followings:

$$S = \text{ctrl} \cdot R + \overline{\text{ctrl}} \cdot S = \text{MUX}(R, S, \text{ctrl}) \tag{4}$$

$$V = \text{ctrl} \cdot U + \overline{\text{ctrl}} \cdot V = \text{MUX}(U, V, \text{ctrl}). \tag{5}$$

Based on the above, Fig. 1 shows the sequential division architecture and its detailed architecture for control unit. We note that in the initial step, $R = B$, $S = G(x)$, $U = A$, $V = 0$, $\beta = -1$. Applying Algorithm 1, the division has the latency of $2m - 1$ clock cycles.

B. Single Parity Code (SPC)

In coding theory, $(m + 1, m)$ SPC [36] includes m-bit message and one-bit parity. Assume that $U = u_0 + u_1 x + \cdots + u_{m-1}x^{m-1}$ is an m-bit message, the codeword C using the SPC representation is then constructed as $C = P_U + xU$, where P_U is the parity of U defined as follows:

$$P_U = u_0 + u_1 + \cdots + u_{m-1} \bmod 2. \tag{6}$$

We use P and $\hat{P}$ for the actual and predicted parities, respectively. Let the message U correspond to the field element, then, we have the following:

1) if $g \in$ GF(2), then $\hat{P}_{gU} = gP_U$;
2) if A and B are two elements in GF(2^m), then $\hat{P}_{A+B} = P_A + P_B$;
3) assuming the irreducible polynomial $G(x) = g_0 + g_1 x + \cdots + g_{(m-1)}x^{(m-1)} + x^m$ is used to generate the finite field GF(2^m). Then, we can obtain $\hat{P}_{xA \bmod G(x)} = P_A + a_{m-1}$, with $A = a_0 + a_1 x + \cdots + a_{m-1}x^{m-1}$.

As stated above, the SPC is capable of detecting odd number of errors in the codewords. Note that applying SPC, CED multipliers over GF(2^m) are presented in [32]–[34].

III. Proposed Three-Type Parity Prediction Schemes

The division algorithm implementations presented in the previous section can be used in high-performance applications. Nevertheless, these architectures are not able to detect the errors occurring in their hardware implementations to achieve error-free division results. To realize reliable architectures and as a step-forward toward thwarting fault analysis attacks in the cryptographic implementations using this algorithm, in this section, we propose three-type parity prediction schemes for implementing error detection division architectures.

A. Definition of Parity Prediction Representation

Let us denote the ith recursive division result as $C = c_0 + c_1x + \cdots + c_{m-1}x^{m-1}$ and let us select the value n as $n = \lceil m/k \rceil$. Then, C can be represented as an $n \times k$ matrix whose elements are the coefficients of the ith recursive division except for $n - \lceil m/k \rceil$ elements which are zero (the coefficient vector is zero-padded). In other words, jth coefficients of the polynomial for $m < j <$ nk are zero. Each parity bit is the sum of k-term coefficients. Therefore, the proposed CED structure is called n-multiple parity prediction scheme. For $n = 1$, the proposed CED scheme is called single parity prediction scheme. For the sake of regularity and complying with the general scheme, the coefficients can be also arranged by an $n \times n$ array structure, where $n = \lceil \sqrt{m} \rceil$. According to (6), we define three parity types, i.e., row parity $P_C^{(0)} = (P_{C_0}^{(0)}, P_{C_1}^{(0)}, \ldots, P_{C_{n-1}}^{(0)})$, column parity $P_C^{(1)} = (P_{C_0}^{(1)}, P_{C_1}^{(1)}, \ldots, P_{C_{n-1}}^{(1)})$, and cross parity $P_C^{(2)} = (P_{C_0}^{(2)}, P_{C_1}^{(2)}, \ldots, P_{C_{n-1}}^{(2)})$. Each of these parity bits is defined as follows:

$$P_{C_j}^{(0)} = \sum_{l=0}^{n-1} c_{jn+l}, \quad 0 \le j \le n-1 \tag{7}$$

$$P_{C_j}^{(1)} = \sum_{l=0}^{n-1} c_{nl+j}, \quad 0 \le j \le n-1 \tag{8}$$

$$P_{C_j}^{(2)} = \sum_{l=0}^{n-j-1} c_{(n+1)l+j} + \sum_{l=n-j}^{n-1} c_{(n+1)l+j-n}, \ 0 \le j \le n-1. \tag{9}$$

For clarity, we use the following example to illustrate the derivations for our presented three parities.

Example 1: Let $C = \sum_{i=0}^{8} c_i x^i$, then, it can be arranged in the following 3×3 array:

$$C = \begin{pmatrix} c_0 & c_1 & c_2 \\ c_3 & c_4 & c_5 \\ c_6 & c_7 & c_8 \end{pmatrix}.$$

According to (7)–(9), we can obtain three parities, such that the row parity $P_C^{(0)} = (c_0+c_1+c_2, c_3+c_4+c_5, c_6+c_7+c_8)$, the column parity $P_C^{(1)} = (c_0+c_3+c_6, c_1+c_4+c_7, c_2+c_5+c_8)$, and the cross parity $P_C^{(2)} = (c_0+c_4+c_8, c_1+c_5+c_6, c_2+c_3+c_7)$.

For the ith iteration result $C^{(i)} = c_0 + c_1x + \cdots + c_{m-1}x^{m-1}$, according to Algorithm 1 and denoting $C^{(i-1)}$ as the $i-1$th iteration, we have $C^{(i)} = C^{(i-1)}/x \mod G(x)$. Considering the irreducible polynomial $G(x)$, it is straightforward to obtain $C^{(i)} = c_{i-1,0} + \cdots + c_{i-1,m-1}x^{m-2} + c_{i-1,0}(g_1 + \cdots + g_{m-1}x^{m-2} + x^{m-1})$. Let us denote $c_{i-1,0} + \cdots + c_{i-1,m-1}x^{m-2} + c_{i-1,0}x^{m-1}$ as $\overline{C}$, then, we have $\overline{C} = C^{(i-1)}/x \mod (x^m + 1)$.

Proposition 1: Let the $i-1$th recursive division result be denoted by C (for the sake of brevity) and be arranged by an $n \times n$ array structure, where $n = \lceil \sqrt{m} \rceil$. Based on (7)–(9), let us define three parity polynomials $P_C^{(0)}$, $P_C^{(1)}$, and $P_C^{(2)}$ as $P_{C_0}^{(i)} + P_{C_1}^{(i)}x + \cdots + P_{C_{n-1}}^{(i)}x^{n-1}$, $0 \le i \le 2$, respectively (for simplicity, we use similar notations for parity vectors and polynomials). Then, three predicted parities of $\overline{C}$ are formed by the following:

$$\widehat{P}_{\overline{C}}^{(0)} = P_C^{(0)} + W^{(-1)} + W \tag{10}$$

$$\widehat{P}_{\overline{C}}^{(1)} = x^{-1}P_C^{(1)} \mod (x^n + 1) \tag{11}$$

$$\widehat{P}_{\overline{C}}^{(2)} = x^{-1}P_C^{(2)} \mod (x^n + 1) + \overline{W}^{(1)} + \overline{W} \tag{12}$$

where

$$W = w_0 + w_1x + \cdots + w_{n-1}x^{n-1} \tag{13}$$

$$w_i = c_{ni}, \quad \text{for} \ \ 0 \le i \le n-1$$

$$W^{(i)} = x^i W \mod (x^n + 1) \tag{14}$$

$$\overline{W} = w_{n-1} + w_{n-2}x + \cdots + w_0x^{n-1}. \tag{15}$$

Proof: We present the proof for row parity, noting that the results for column and cross parities can be very similarly obtained. Considering the parity polynomials, the row parity $\widehat{P}_{\overline{C}}^{(0)}$ is derived as follows:

$$\begin{aligned} \widehat{P}_{\overline{C}}^{(0)} &= (c_0 + c_n + P_{C_0}^{(0)}) + (c_n + c_{2n} + P_{C_1}^{(0)})x \\ &\quad + \cdots + (c_{(n-1)n} + c_0 + P_{C_{n-1}}^{(0)})x^{n-1} \\ &= P_{C_0}^{(0)} + P_{C_1}^{(0)}x + \cdots + P_{C_{n-1}}^{(0)}x^{n-1} \\ &\quad + c_n + c_{2n}x + \cdots + c_0x^{n-1} \\ &\quad + c_0 + c_nx + \cdots + c_{(n-1)n}x^{n-1} \\ &= P_C^{(0)} + W^{(-1)} + W. \end{aligned}$$

■

In the following, based on the three parity presentations in (7)–(9), we derive a novel parity prediction scheme that is used in the presented fault-immune architectures.

B. Parity Prediction for $C^{(i)} = C^{(i-1)}/x \ mod \ G(x)$

For computing the parity prediction for $C^{(i)} = C^{(i-1)}/x \mod G(x)$ operation, the column-based type parity prediction is presented as follows (those for row and cross parities are obtained in a similar manner).

Predicted Column Parity $\widehat{P}_{C^{(i)}}^{(1)}$: Two column parity polynomials for G and $C^{(i)}$ are represented by

$$P_G^{(1)} = P_{G_0}^{(1)} + P_{G_1}^{(1)}x + \cdots + P_{G_{n-1}}^{(1)}x^{n-1} \tag{16}$$

$$P_{C^{(i)}}^{(1)} = P_{C_0^{(i)}}^{(1)} + P_{C_1^{(i)}}^{(1)}x + \cdots + P_{C_{n-1}^{(i)}}^{(1)}x^{n-1} \tag{17}$$

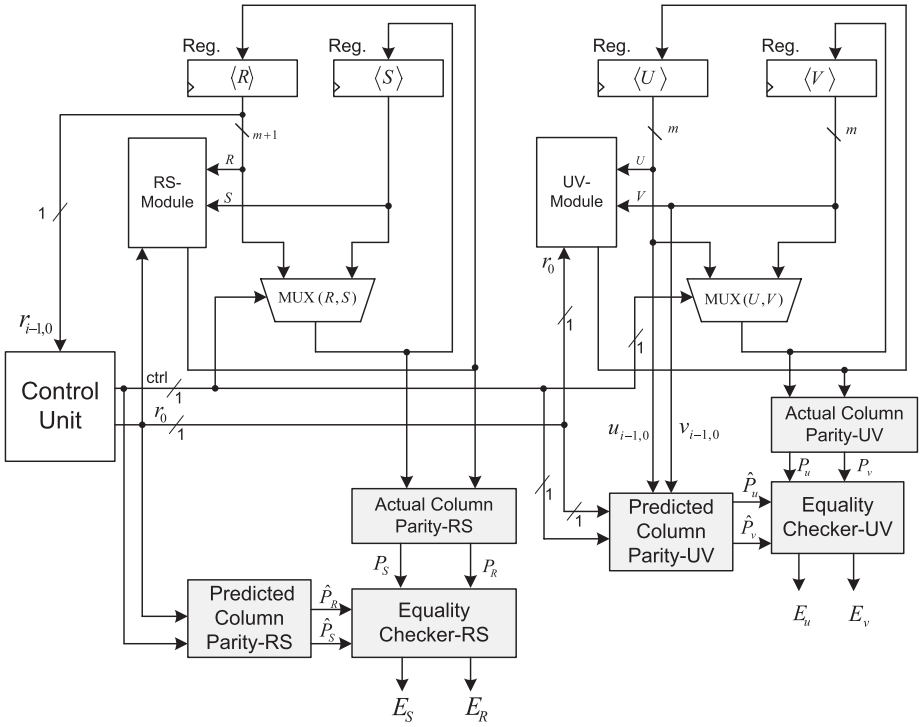

Fig. 2. Proposed error detection architecture for the EEA over GF(2^m) using the column parity prediction scheme.

where

$$P^{(1)}_{G_j} = \sum_{l=0}^{n-1} g_{l\times n+j+1} \quad \text{for} \quad 0 \le j \le n-1$$

$$P^{(1)}_{C_j} = \sum_{l=0}^{n-1} c_{i,l\times n+j} \quad \text{for} \quad 0 \le j \le n-1.$$

Then, the predicted column parity of $C^{(i)}$ is represented by the following:

$$\widehat{P}^{(1)}_{C^{(i)}} = P^{(1)}_{\overline{C}} + c_{i-1,0} P^{(1)}_{\overline{G}}. \tag{18}$$

Similarly, based on Proposition 1 (see the result for column parity) and (18), and considering $n = \lceil \sqrt{m} \rceil$, for computing the predicted parity $\widehat{P}^{(1)}_{\overline{C}}$ in a recursive manner, we have the following:

$$\widehat{P}^{(1)}_{\overline{C}} = x^{-1} \widehat{P}^{(1)}_{C^{(i-1)}} \mod (x^n + 1). \tag{19}$$

Thus, the predicted column parity $\widehat{P}^{(1)}_{C^{(i)}}$ is obtained as follows:

$$\widehat{P}^{(1)}_{C^{(i)}} = x^{-1} \widehat{P}^{(1)}_{C^{(i-1)}} \mod (x^n + 1) + c_{i-1,0} P^{(1)}_{\overline{G}}. \tag{20}$$

C. Parity Prediction for MUX($U^{(i-1)}$, $V^{(i-1)}$, ctrl)

Note $V^{(i)} = \text{MUX}(U^{(i-1)}, V^{(i-1)}, \text{ctrl})$ represents $V^{(i)} = U^{(i-1)} \cdot \text{ctrl} + V^{(i-1)} \cdot \overline{\text{ctrl}}$. That is, the value of V depends on the binary value of ctrl. Assume that $P_{U^{(i-1)}}$ and $P_{V^{(i-1)}}$ are the parities of $U^{(i-1)}$ and $V^{(i-1)}$, respectively. Then, the parity prediction of $V^{(i)}$ is performed as follows:

$$\begin{aligned} \widehat{P}_{V^{(i)}} &= P_{U^{(i-1)}} \cdot \text{ctrl} + P_{V^{(i-1)}} \cdot \overline{\text{ctrl}} \\ &= \text{MUX}(P_{U^{(i-1)}}, P_{V^{(i-1)}}, \text{ctrl}). \end{aligned} \tag{21}$$

IV. Proposed CED Division Architecture

In the previous section, we presented parity prediction derivations for individual modules in our proposed scheme. According to the division architecture in Fig. 1, this section presents the proposed CED division architecture for the column-based scheme, those for row and cross parities are presented subsequently through a remark.

Based on Section II-A, let us denote $R^{(i)}$,$S^{(i)}$,$U^{(i)}$, and $V^{(i)}$ as the ith iterative results. In addition, in Algorithm 1, we have $s_{i-1,0} = 1$ and $r_0 = r_{i-1,0}$ for each iterative operation. Assume that two predicted column parities $\widehat{P}^{(1)}_{R^{(i-1)}}$ and $\widehat{P}^{(1)}_{S^{(i-1)}}$ are pre-calculated. Based on (1), the predicted column parity $\widehat{P}^{(1)}_{R^{(i)}}$ using (20) can be obtained as follows:

$$\begin{aligned} \widehat{P}^{(1)}_{R^{(i)}} = \frac{(\widehat{P}^{(1)}_{R^{(i-1)}} + r_0 \widehat{P}^{(1)}_{S^{(i-1)}})}{x} \mod (x^n + 1) \\ + (r_{i-1,0} + r_0 s_{i-1,0}) P^{(1)}_{\overline{G}}. \end{aligned}$$

According to (4), the predicted column parity $\widehat{P}^{(1)}_{S^{(i)}}$ using (21) can be obtained by the following:

$$\widehat{P}^{(1)}_{S^{(i)}} = \text{MUX}(\widehat{P}^{(1)}_{R^{(i-1)}}, \widehat{P}^{(1)}_{S^{(i-1)}}, \text{ctrl}). \tag{22}$$

In addition, two predicted column parities $\widehat{P}^{(1)}_{U^{(i)}}$ and $\widehat{P}^{(1)}_{V^{(i)}}$ are computed as follows:

$$\begin{aligned} \widehat{P}^{(1)}_{U^{(i)}} = \frac{(\widehat{P}^{(1)}_{U^{(i-1)}} + r_0 \widehat{P}^{(1)}_{V^{(i-1)}})}{x} \mod (x^n + 1) \\ + (u_{i-1,0} + r_0 v_{i-1,0}) P^{(1)}_{\overline{G}} \end{aligned} \tag{23}$$

$$\widehat{P}^{(1)}_{V^{(i)}} = \text{MUX}(\widehat{P}^{(1)}_{U^{(i-1)}}, \widehat{P}^{(1)}_{V^{(i-1)}}, \text{ctrl}). \tag{24}$$

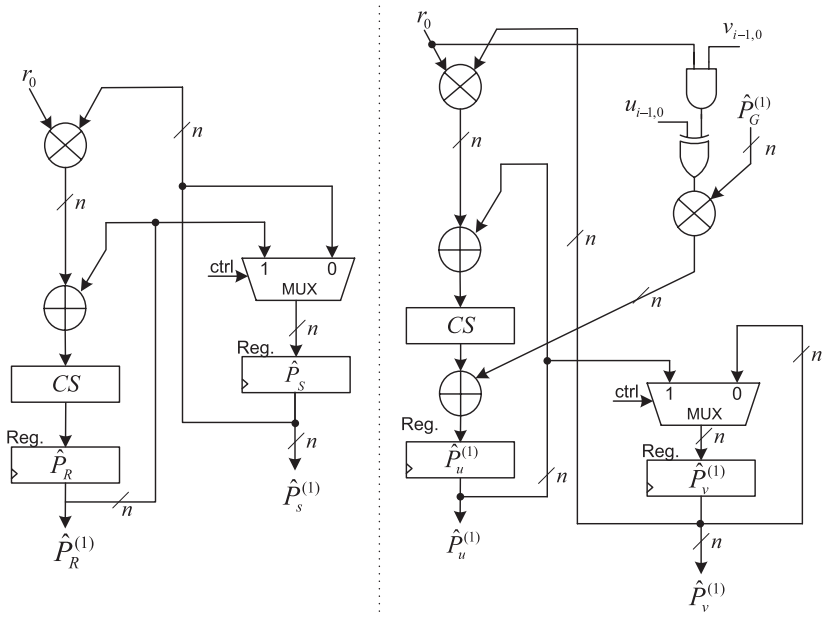

Fig. 3. Column parity prediction architectures.

For clarity, we use the column parity prediction scheme to illustrate our proposed CED EEA division architecture as shown in Fig. 2. In Fig. 2, the Actual Column Parity-RS module computes two actual column parities $P_R^{(1)}$ and $P_S^{(1)}$. In addition, that of UV computes two actual column parities $P_U^{(1)}$ and $P_V^{(1)}$. The Predicted Column Parity-RS module computes two predicted column parities $\widehat{P}_R^{(1)}$ and $\widehat{P}_S^{(1)}$, respectively, as shown in Fig. 3. Additionally, the two predicted column parities $\widehat{P}_U^{(1)}$ and $\widehat{P}_V^{(1)}$ are obtained as shown in this figure. The equality checker module performs $E_R = P_R + \widehat{P}_R$, $E_S = P_S + \widehat{P}_S$, $E_U = P_U + \widehat{P}_U$, and $E_V = P_V + \widehat{P}_V$ operations. Hence, we are able to use four indicators to detect division computation errors. We note that after parity predictions, the CED division architecture requires one extra clock cycle to compute the error checking indicators (E_R, E_S, E_U, E_V). The proposed CED architecture detects the occurrences of permanent or transient errors in the division architecture if any of four indicators becomes nonzero.

Remark 1: One can obtain the row parity prediction results through the following formulae, the proof of which is not presented for the sake of brevity

$$\begin{aligned}
\widehat{P}^{(0)}_{R^{(i)}} &= \widehat{P}^{(0)}_{R^{(i-1)}} + r_0\widehat{P}^{(0)}_{S^{(i-1)}} + W^{(-1)}_{\mathrm{RS}} + W^{(0)}_{\mathrm{RS}} \\
\widehat{P}^{(0)}_{U^{(i)}} &= \widehat{P}^{(0)}_{U^{(i-1)}} + r_0\widehat{P}^{(0)}_{V^{(i-1)}} \\
&\quad +(u_{i-1,0} + r_0 v_{i-1,0})P^{(0)}_{\overline{G}} + W^{(-1)}_{\mathrm{UV}} + W^{(0)}_{\mathrm{UV}} \\
\widehat{P}^{(0)}_{S^{(i)}} &= \mathrm{MUX}(\widehat{P}^{(0)}_{R^{(i-1)}}, \widehat{P}^{(0)}_{S^{(i-1)}}, \mathrm{ctrl}) \\
\widehat{P}^{(0)}_{V^{(i)}} &= \mathrm{MUX}(\widehat{P}^{(0)}_{U^{(i-1)}}, \widehat{P}^{(0)}_{V^{(i-1)}}, \mathrm{ctrl})
\end{aligned} \tag{25}$$

where

$$W_{\mathrm{RS}} = \sum_{i=0}^{n-1}(r_{i-1,ni} + s_{i-1,ni})x^i$$

$$W_{\mathrm{UV}} = \sum_{i=0}^{n-1}(u_{i-1,ni} + v_{i-1,ni})x^i.$$

In addition, one can obtain the cross parity prediction results as follows:

$$\begin{aligned}
\widehat{P}^{(2)}_{R^{(i)}} &= \widehat{P}^{(2)}_{R^{(i-1)}} + r_0\widehat{P}^{(2)}_{S^{(i-1)}} + \overline{W}^{(-1)}_{\mathrm{RS}} + \overline{W}^{(0)}_{\mathrm{RS}} \\
\widehat{P}^{(2)}_{U^{(i)}} &= \widehat{P}^{(2)}_{U^{(i-1)}} + r_0\widehat{P}^{(2)}_{V^{(i-1)}} \\
&\quad +(u_{i-1,0} + r_0 v_{i-1,0})P^{(2)}_{\overline{G}} + \overline{W}^{(-1)}_{\mathrm{UV}} + \overline{W}^{(0)}_{\mathrm{UV}} \\
\widehat{P}^{(2)}_{S^{(i)}} &= \mathrm{MUX}(\widehat{P}^{(2)}_{R^{(i-1)}}, \widehat{P}^{(2)}_{S^{(i-1)}}, \mathrm{ctrl}) \\
\widehat{P}^{(2)}_{V^{(i)}} &= \mathrm{MUX}(\widehat{P}^{(2)}_{U^{(i-1)}}, \widehat{P}^{(2)}_{V^{(i-1)}}, \mathrm{ctrl})
\end{aligned} \tag{26}$$

where

$$\overline{W}_{\mathrm{RS}} = \sum_{i=0}^{n-1}(r_{i-1,ni} + s_{i-1,ni})x^{n-i-1}$$

$$\overline{W}_{\mathrm{UV}} = \sum_{i=0}^{n-1}(u_{i-1,ni} + v_{i-1,ni})x^{n-i-1}.$$

V. Error Simulations and Overhead Benchmark

Here, we present the results of our error simulations to benchmark the effectiveness of the proposed approaches. In addition, time and hardware overheads of the presented schemes are presented.

A. Error Detection Capability

In error control coding theory, multiple parity prediction with n-bit parity, for instance, for implementing the CED multiplier over GF(2^m) [37], has the error coverage of

$$\mathrm{EC}\ (\%) = 100 \times \left(\frac{2^{m+n} - 2^m}{2^{m+n}}\right)\% = 100 \times \left(1 - \frac{1}{2^n}\right)\%.$$

TABLE I
ERROR DETECTION CAPABILITY OF OUR PROPOSED CED DIVISION ARCHITECTURE FOR STUCK-AT-1 (STUCK-AT-0) FAULT MODELS

Number of errors	Type	Parity bits				
		1	3	5	7	9
Four (multiple)	Injected	9949 (7188)	9958 (7220)	9958 (7182)	9946 (7117)	9957 (7258)
	Detected	8869 (6029)	9671 (6850)	9794 (6970)	9844 (6980)	9880 (7179)
	EC (%)	89.14 (83.88)	97.11 (94.88)	98.35 (97.05)	98.97 (98.08)	99.22 (98.91)
Three (multiple)	Injected	9835 (6239)	9802 (6259)	9787 (6198)	9815 (6222)	9788 (6241)
	Detected	8668 (5335)	9420 (5970)	9590 (6028)	9667 (6108)	9666 (6157)
	EC (%)	88.13 (85.51)	96.1 (95.38)	97.98 (97.26)	98.49 (98.17)	98.75 (98.65)
Two (multiple)	Injected	9242 (4830)	9227 (4812)	9235 (4922)	9264 (4838)	9244 (4870)
	Detected	7808 (4220)	8817 (4595)	8975 (4806)	9075 (4766)	9104 (4817)
	EC (%)	84.48 (87.37)	95.55 (95.49)	97.18 (97.64)	97.95 (98.51)	98.48 (98.91)
One (single)	Injected	7331 (2872)	7317 (2781)	7269 (2768)	9296 (2800)	7322 (2777)
	Detected	7331 (2872)	7317 (2781)	7269 (2768)	9296 (2800)	7322 (2777)
	EC (%)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)

To evaluate the error detection capability of our proposed CED division architecture, we use MATLAB to simulate the EEA division over GF(2^{163}). We select the field-defining polynomial from the National Institute of Standards and Technology recommended polynomials for Elliptic Curve Digital Signature Algorithm as $x^{163} + x^7 + x^6 + x^3 + 1$. Let us consider the CED architecture using column parity prediction in Fig. 2. Then, in the initial step, we consider four fault injection cases to inject faults for four registers (R, S, U, V) using 10 000 random inputs. We select five parity-bit types to obtain the error detection capability of our proposed architecture. Table I shows our results based on our fault models. As seen in this table, the proposed CED architecture can reach 100% error coverage for single stuck-at-1 (stuck-at-0) fault models. As the number of injected faults goes beyond three, the proposed CED architecture with seven parity bits can achieve about 99% error coverage. Note that the results of our experiments are for both transient and permanent faults that can potentially cover both natural faults and malicious fault analysis. In addition, the location and number of injections are randomly chosen.

Based on the structure of Fig. 2, our proposed CED architecture uses dual multiple parity prediction scheme. In this regard, we have the following: 1) given our proposed CED architecture, it is shown that all the single-bit errors in the CED divider architecture can be detected and 2) from the error simulation results, it is derived that the error detection capability is more than 97% if the selected parity length is at least five bits (using dual multiple parity scheme). We obtain very close to 100% error coverage by increasing the number of parity bits as shown through the simulation results in Table I, according to reliability objectives and in case of overhead tolerance.

B. Time and Hardware Overheads

In this subsection, we present the time and hardware overheads of the proposed error detection scheme (including equality checkers) for the EEA division. The time and area complexity benchmark of the three predicted parity schemes shows that two parity prediction schemes, i.e., row parity scheme and cross parity scheme, have the same time and area overheads. The column parity prediction approach, however, saves $6n$ XOR gates compared with the row/cross parity prediction schemes. The time overhead of the column parity prediction scheme is also less by $(2m - 1)T_A$ compared with those for the row/cross parity schemes. Therefore, the column parity prediction scheme has lower time and area overheads as compared with the row/cross parity prediction schemes.

For comparing the time and area overheads, we use the NanGate standard-cell library[1] [38]. We use the typical corner ($V_{\mathrm{dd}} = 1.10$ V and $T_j = 25$ °C) and the drive strength of one for all the utilized primitives in this subsection (implying similar input transition and load capacitance for the primitives). Based on these, as seen in Table II, we have presented the time and area overheads for the proposed CED division architecture over GF(2^{163}) using the parity prediction schemes for eight parity bits and also the time and area overheads for various number of parity bits. As shown in this table and for eight parity bits, without parity prediction scheme, we have the area of 6330 μm^2 and the critical path delay of 71.5 ns and with parity prediction for the column-based approach, these are 7611 μm^2 and 71.74 ns, respectively. Therefore, the area and the time overheads of 20.24% and 0.33% are achieved as seen in this table for the column-based approach. In addition, as shown in Table II and for 8 parity bits, these are 7688 μm^2 and 78.20 ns for the row/cross parity-based architecture. Therefore, the area and time overheads of 21.45% and 9.37% for row/cross-based parity schemes are achieved.

As shown in Table II, we have also presented the overhead results for seven different parity bits, i.e., for 2, 4, 6, 8, 10, 12, and 14 parity bits. These correspond to the area overheads of 17.72% to 25.18% for row/cross parity-based scheme and 17.42% to 23.07% for the column-based scheme. In other words, increasing the number of parity bits increases the area overheads at the gain of higher error coverage. As seen in

[1]The library was generated using NanGate's Library Creator and the 45-nm FreePDK Base Kit from North Carolina State University (NCSU) and characterization was done using the Predictive Technology Model (PTM) from Arizona State University (ASU) [38].

TABLE II
TIME AND AREA OVERHEADS FOR THE PROPOSED CED DIVISION ARCHITECTURE OVER GF(2^{163}) USING THE PRESENTED PARITY PREDICTION SCHEMES

(a) Overheads for the presented CED division architecture using 8 parity bits.

Divider architecture	Area (μm^2)	Delay (ns)	Overhead(%)	
			Area	Time
Without parity prediction scheme	6330	71.50	–	–
With row/cross parity prediction scheme	7688	78.20	21.45	9.37
With column parity prediction scheme	7611	71.74	20.24	0.33

(b) Area and time overheads for various number of parity bits.

Scheme	# Parity bits	2	4	6	8	10	12	14
Row/cross parity	Area overhead (%)	17.72	18.96	20.21	21.45	22.70	23.94	25.18
	Time overhead (%)	9.53	9.48	9.42	9.37	9.37	9.37	9.37
Column parity	Area overhead (%)	17.42	18.36	19.30	20.24	21.18	22.12	23.07
	Time overhead (%)	0.44	0.39	0.33	0.33	0.33	0.27	0.27

this table, the time overheads for these parity bits are at most 9.53% (for the row/cross-based scheme) and 0.44% for the column-based approach. The area and time overheads shown in Table II can be used to get insight about the effectiveness of the proposed scheme. For instance, for column-based (row/cross-based) scheme and for eight parity bits, we reach the area and time overheads of 20.24% (21.45%) and 0.33% (9.37%), respectively. However, the results presented in [34] with CED for multiplications reaches more than 40% and 20% area and time overheads, respectively, for the same number of parity bits (this is observation and not comparison, as the architectures are different). Note that based on the reliability constraints and the available resources, one can choose the schemes presented in this paper with the benchmarks presented in this section to have more reliable EEA hardware architectures.

VI. CONCLUSION

In this paper, we presented three parity prediction schemes to make the EEA division algorithm more reliable. The proposed CED division architecture was formed by dual multiple parity prediction approaches. In this regard, it was shown that the column parity prediction scheme had lower time and space overheads compared with the row/cross parity prediction schemes. Through our simulations, it was shown that the CED division architecture using dual multiple parity prediction scheme was capable of reaching close to 100% error coverage for single and multiple stuck-at faults. This made the architectures for the EEA division scheme more reliable while keeping their high performance characteristics by having acceptable time$\times$area overheads. One may choose the proposed fault-immune architectures to reach the reliability objectives needed, depending on the resources available and the performance and efficiency to reach for sensitive applications such as those utilizing the ECC architectures for providing various security mechanisms such as establishing key agreement schemes, augmented encryption approaches, and digital signature algorithms.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their constructive comments.

REFERENCES

[1] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*. New York, NY, USA: Cambridge Univ. Press, 1999.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] G. Orlando and C. Paar, "A high-performance reconfigurable elliptic curve processor for GF(2^m)," in *Proc. Workshop Cryptograph. Hardw. Embedded Syst.*, 2000, pp. 31–43.

[4] A. Royo, J. Morán, and J. C. López, "Design and implementation of a coprocessor for cryptography applications," in *Proc. Eur. Design Test Conf. Proc.*, 1997, pp. 213–217.

[5] J. Adikari, V. S. Dimitrov, and R. J. Cintra, "A new algorithm for double scalar multiplication over Koblitz curves," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2011, pp. 709–712.

[6] R. Farashahi and M. Joye, "Efficient arithmetic on Hessian curves," in *Proc. Int. Conf. Pract. Theory Public Key Cryptography*, 2010, pp. 243–260.

[7] S.-W. Wei, "VLSI architectures for computing exponentiations, multiplicative inverses, and divisions in GF(2^m)," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 44, no. 10, pp. 847–855, Oct. 1997.

[8] J. H. Guo and C. L. Wang, "Systolic array implementation of Euclid's algorithm for inversion and division in GF(2^m)," *IEEE Trans. Comput.*, vol. 47, no. 10, pp. 1161–1167, Oct. 1998.

[9] C. H. Wu, C. M. Wu, M. D. Shieh, and Y. T. Hwang, "High-speed, low-complexity systolic designs of novel iterative division algorithms in GF(2^m)," *IEEE Trans. Comput.*, vol. 53, no. 3, pp. 375–380, Mar. 2004.

[10] J. H. Guo and C. L. Wang, "Hardware-efficient systolic architecture for inversion and division in GF(2^m)," *IEE Comput. Digit. Tech.*, vol. 145, no. 4, pp. 272–278, Jul. 1998.

[11] N. Takagi, "A VLSI algorithm for modular division based on the binary GCD algorithm," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E81-A, no. 5, pp. 724–728, May 1998.

[12] K. Kobayashi and N. Takagi, "Fast hardware algorithm for division in GF(2^m) based on the extended Euclid's algorithm with parallelization of modular reductions," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 8, pp. 644–648, Aug. 2009.

[13] D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults," in *Proc. Eurocrypt*, 1997, pp. 37–51.

[14] I. Biehl, B. Meyer, and V. Muller, "Differential fault attacks on elliptic curve cryptosystems," in *Proc. 20th Annu. Int. Cryptol. Conf.*, 2000, pp. 131–146.

[15] J. Blomer and J. P. Seifert, "Fault based cryptanalysis of the advanced encryption standard (AES)," in *Proc. Financial Cryptography*, Jan. 2003, pp. 162–181.

[16] J. Blomer and V. Krummel, "Fault based collision attacks on AES," in *Proc. Int. Workshop Fault Diagnosis Tolerance Cryptography*, Oct. 2006, pp. 106–120.

[17] J. Takahashi, T. Fukunaga, and K. Yamakoshi, "DFA mechanism on the AES key schedule," in *Proc. Int. Workshop Fault Diagnosis Tolerance Cryptography*, Sep. 2007, pp. 62–72.

[18] C. H. Yen and B .F. Wu, "Simple error detection methods for hardware implementation of advanced encryption standard," *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 720–731, Jun. 2006.

[19] T. G. Malkin, F. X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Proc. Int. Workshop Fault Diagnosis Tolerance Cryptography*, Oct. 2006, pp. 159–172.

[20] G. Di Natale, M. Doulcier, M. L. Flottes, and B. Rouzeyre, "A reliable architecture for parallel implementations of the advanced encryption standard," *J. Electron. Test., Theory Appl.*, vol. 25, no. 4, pp. 269–278, Aug. 2009.

[21] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 608–622, May 2010.

[22] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight high-performance fault detection scheme for the advanced encryption standard using composite fields," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 1, pp. 85–91, Jan. 2011.

[23] A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "High-performance concurrent error detection scheme for AES hardware," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2008, pp. 100–112.

[24] P. Maistri and R. Leveugle, "Double-data-rate computation as a countermeasure against fault analysis," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1528–1539, Nov. 2008.

[25] M. Mozaffari-Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron.*, [Online]. Available: http://dx.doi.org/10.1109/TIE.2012.2228144.

[26] P. K. Meher, "Systolic and non-systolic scalable modular designs of finite field multipliers for reed-solomon codec," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 6, pp. 747–757, Jun. 2009.

[27] P. K. Meher, "Systolic and super-systolic multipliers for finite field GF(2^m) based on irreducible trinomials," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 4, pp. 1031–1040, May 2008.

[28] J. Xie, P. K. Meher, and J. He, "Low-complexity multiplier for GF(2^m) based on all-one polynomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 168–173, Jan. 2013.

[29] J. Adikari, A. Barsoum, M. A. Hasan, A. H. Namin, and C. Negre, "Improved area-time trade-offs for field multiplication using optimal normal bases," *IEEE Trans. Comput.*, vol. 62, no. 1, pp. 193–199, Jan. 2013.

[30] C. H. Kim, C. P. Hong, and S. Kwon, "A digit-serial multiplier for finite field GF(2^m)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 4, pp. 476–483, Apr. 2005.

[31] C. Y. Lee, C. W. Chiou, and J. M. Lin, "Concurrent error detection in polynomial basis multiplier over GF(2^m)," *J. Electron. Test., Theory Appl.*, vol. 22, pp. 143–50, Feb. 2006.

[32] S. Fenn, M. Gossel, M. Benaissa, and D. Taylor, "On-line error detection for bit-serial multipliers in GF(2^m)," *J. Electron. Test., Theory Appl.*, vol. 13, no. 1, pp. 29–40, 1998.

[33] C. Y. Lee, C. W. Chiou, and J. M. Lin, "Concurrent error detection in a bit-parallel systolic multiplier for dual basis of GF(2^m)," *J. Electron. Test., Theory Appl.*, vol. 21, pp. 539–549, Sep. 2005.

[34] S. Bayat-Sarmadi and M. Anwar Hasan, "On concurrent detection of errors in polynomial basis multiplication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 4, pp. 413–426, Apr. 2007.

[35] Y.-C. Chuang and C.-W. Wu, "On-line error detection schemes for a systolic finite-field inverter," in *Proc. Asian Test Symp.*, 1998, pp. 301–305.

[36] S. Lin and D. J. Costello, *Error Control Coding*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[37] W. Chelton and M. Benaissa, "Concurrent error detection in GF(2^m) multiplication and its application in elliptic curve cryptography," *IET Circuits, Devices Syst.*, vol. 2, no. 3, pp. 289–297, 2008.

[38] (2013). *45-nm FreePDK Base Kit* [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib/

Mehran Mozaffari-Kermani (M'11) received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2005, and the M.E.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada, in 2007 and 2011, respectively.

He joined the Advanced Micro Devices as a Senior ASIC/Layout Designer, integrating sophisticated security/cryptographic capabilities into a single accelerated processing unit. In 2012, he joined the Electrical Engineering Department, Princeton University, Princeton, NJ, USA, as an NSERC Post-Doctoral Research Fellow. As of 2013, he has been an Assistant Professor with the Department of Electrical and Microelectronic Engineering at Rochester Institute of Technology, Rochester, NY, USA. His current research interests include emerging security/privacy measures for deeply embedded systems, cryptographic hardware systems, fault diagnosis and tolerance in cryptographic hardware, VLSI reliability, and low-power secure and efficient FPGA and ASIC designs.

Dr. Mozaffari-Kermani is a member of the IEEE Computer Society. He was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellowship in 2011.

Reza Azarderakhsh received the B.Sc. degree in electrical and electronic engineering in 2002, the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2005, and the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2011.

He joined the Department of Electrical and Computer Engineering, University of Western Ontario, as a Limited Duties Instructor, in September 2011. He has been a Post-Doctoral Fellow with the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. As of 2013, he has been an Assistant Professor with the Department of Computer Engineering at Rochester Institute of Technology, Rochester, NY, USA. His current research interests include finite field and its application, elliptic curve cryptography, and pairing based cryptography.

Chiou-Yng Lee (SM'08) received the Bachelor's degree in medical engineering and the M.S. degree in electronic engineering from Chung Yuan Christian University, Jhongli, Taiwan, in 1986 and 1992, respectively, and the Ph.D. degree in electrical engineering from Chang Gung University, Taiwan, in 2001.

He was a Research Associate with the Chunghwa Telecommunication Laboratory, Taiwan, from 1988 to 2005. He joined the Department of Project Planning. He taught those related field courses with Ching Yun University. Currently, he is a Professor with the Department of Computer Information and Network Engineering, Lunghwa University of Science and Technology. His current research interests include computations in finite fields, error-control coding, signal processing, and digital transmission system.

Dr. Lee is a senior member of the IEEE Computer Society. He is an Honor Member of Phi Tao Phi in 2001.

Siavash Bayat-Sarmadi (M'08) received the B.Sc. degree from the University of Tehran, Tehran, Iran, in 2000, the M.Sc. degree from the Sharif University of Technology, Tehran, in 2002, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2007, all in computer engineering (hardware).

In 2007, he joined Advanced Micro Devices, Inc. As of 2013, he has been an Assistant Professor with the Department of Computer Engineering at Sharif University of Technology, Tehran, Iran. His current research interests include computer arithmetic and architecture, cryptographic hardware and embedded systems, and fault-tolerant computing.