Applied Mathematics & Information Sciences
An International Journal

Concurrent All-Cell Error Detection in Semi-Systolic Multiplier Using Linear Codes

Wangjie Qiu[1], *Xiao Zhang*[1], *Haoyang Li*[1], *Zhao Wang*[1], *Yao Zhang*[1] *and Zhiming Zheng*[1]

[1]Key Laboratory of Mathematics, Informatics and Behavioral Semantics, Ministry of Education and School of Mathematics and Systems Science, Beihang University, Beijing 100191, P.R. China

Received: 22 Nov. 2012, Revised: 1 Jan. 2013, Accepted: 10 Jan. 2013
Published online: 1 May 2013

Abstract: Finite field multiplier is widely applied in many domains such as coding theory and cryptography. In this paper, it is purposed to propose a concurrent all-cell error detection semi-systolic polynomial basis multiplier based on coding theory which can realize high-speed calculation and high efficient error detection with low resource consumption. First, our method is created to choose an appreciate generator polynomial for a linear code and an irreducible polynomial generating the finite field. Then the finite field arithmetic multiplication with error detection is simplified on a residue class ring resulted from linearly coding. Second, a semi-systolic array is compressed to realize our multiplier which is suitable for almost any finite field with low time and area complexity. Furthermore, our method breaks through the key technical bottleneck of unacceptable time and area overheads in the coding, decoding and checking process. Additionally, the paper creatively builds an all-cell error model for systolic or semi-systolic multipliers more practical than the conventional single stuck-at or single-cell error model. Finally, based on the all-cell error model, the proposed multiplier is only with 2.088% extra time overhead and 4.978% extra area overhead, and its probability of concurrent error detection reaches to 99.999999%.

Keywords: Finite field, linear code, polynomial basis, all-cell error model, semi-systolic multiplier

1. Introduction

Finite field arithmetic plays a fundamental role in some of the most fascinating applications of coding theory, digital signal processing and cryptography [1]. So the design of an efficient multiplier with low time and area complexity is vitally necessary. In general, multipliers are constructed by using polynomial basis (PB) [2], dual basis (DB) [3], and normal basis (NB) [4].

Concurrent error detection (CED) capability of multipliers is required to protect against fault-based attack which injects faults into cryptosystems and leaks secret key information. The most common approaches for error detection include parity prediction, time redundancy, etc. Single parity prediction has been applied to bit-serial multiplier [5], [6], bit-parallel multiplier [7] and Montgomery multiplier [8], [9]. Besides, multiple-bit parity prediction is developed in bit-serial and bit-parallel PB multiplier [10], [11]. Remarkably, in [12], a parity prediction approach is used for a semi-systolic DB multiplier with CED capability.

Time redundancy error detection [13] is created for systolic and semi-systolic multipliers. Based on time redundancy approach, CED method is proposed for PB multiplier [14], [15], [16], DB multiplier [17], and NB multiplier [18]. However, to the best of our knowledge, most CED systolic or semi-systolic multipliers can only detect single stuck-at or single-cell errors. In practice, transient or multiple-cell errors are likely to occur. Therefore, to some sense, the parity prediction and time redundancy approaches for systolic and semi-systolic multipliers are invalid.

A new error detection approach based on error correcting code is raised by Reyhani-Masoleh and Hasan in [19]. Then Gaubatz and Sunar developed this approach by using a class of linear codes according to a generator polynomial [20]. Based on the developed approach, Saramdi and Hasan [21] created a bit-serial and a bit-parallel CED multiplier. A hybrid scheme for CED of multiplication, which combines multiple parity conception together with redundant reduction modules, is raised in [22].

* Corresponding author e-mail: qwj@smss.buaa.edu.cn

In this paper, we construct a semi-systolic PB multiplier by developing the approach described in [21]. Our multiplier, which can be applied to all finite fields, can detect multiple errors and transient errors. Meanwhile, it saves at least 70% time complexity and 63% area complexity compared with other CED systolic or semi-systolic multipliers. Moreover, we break through the key technical bottleneck of high delay for error detection. In addition, an all-cell error model for both systolic and semi-systolic multipliers is built for the first time. Finally, based on the new error model, the error detection capability is analyzed and the probability is strictly proven to be 99.999999% with 2.088% extra time overhead and 4.978% extra area overhead for CED, being more accurate than simulation results.

The remainder of this paper is organized as follows: A brief introduction of finite field and coding theory is presented in Section 2. In Section 3, we propose an algorithm for CED multiplication. In Section 4, a concurrent all-cell error detection semi-systolic PB polynomial multiplier is presented. In Section 5, error model is constructed and error detection capability is analyzed. A comparison is given to evaluate our multiplier in Section 6. Concluding remarks are given in Section 7.

2. Preliminaries

Definition 1.*[1] Let H be an $(n-k)\times n$ matrix of rank $n-k$ with entries in $\mathbb{F}_2$. The set C of all $n-$dimensional vectors $c\in\mathbb{F}_2^n$ such that $Hc^T=0$ is called a binary linear (n,k) code over $\mathbb{F}_2$ where n is called the length and k the dimension of the code. The elements of C are called code words (or code vector). The matrix H is a parity-check matrix of C.*

Theorem 1.*Let $GF(2^m)$ be the finite field of $F(x)$ of degree m. If any polynomial in $GF(2^m)$ is multiplied with a polynomial and arrives at a result set denoted as A, then the vector form of A is called a polynomial code over $\mathbb{F}_2$.*

Theorem 2.*[1] A cyclic (m',m) code C over $\mathbb{F}_2$ can be obtained by multiplying each message (identified with a polynomial of degree $<m$) by a fixed polynomial $g(x)$ of degree n which is a divisor of $x^{m'}+1$, where $m'=m+n$. The parity-check polynomial $h(x)=(x^{m'}-1)/g(x)=\sum_{i=0}^{m}h_ix^i$. Then the matrix*

$$H=\begin{pmatrix} 0 & 0 & \dots & 0 & h_m & h_{m-1} & \dots & h_0 \\ 0 & \dots & 0 & h_m & h_{m-1} & & \dots\ h_0 & 0 \\ \vdots & \vdots & & & & & & \vdots \\ h_m & h_{m-1} & \dots & h_0 & 0 & & \dots & 0 \end{pmatrix} \quad (1)$$

is called the parity-check matrix for C.

3. Algorithm for CED multiplication

3.1. Conventional multiplication algorithm

In this section, we will introduce a conventional multiplication algorithm over $GF(2^m)$ using PB.

Let $F(x)$ be an irreducible polynomial in $\mathbb{F}_2[x]$ of degree m. Then the finite field of $F(x)$ over $\mathbb{F}_2$ is given by $GF(2^m)$. Let α be a root of an irreducible polynomial $F(x)=x^m+\sum_{i=0}^{m-1}p_ix^i$ of degree m. Then the PB is the follow set:$\{1,\alpha,\alpha^2,\alpha^3,\dots,\alpha^{m-2},\alpha^{m-1}\}$. An element A of $GF(2^m)$ can be represented by the PB above as $A=\sum_{i=0}^{m-1}a_i\alpha^i$ where $a_i\in\mathbb{F}_2$. And it can also be viewed as a vector $(a_0,a_1,\dots,a_{m-2},a_{m-1})$.

Suppose A and B are two elements in the binary finite field $GF(2^m)$ where an (primitive) irreducible polynomial $F(x)$ of degree m over $\mathbb{F}_2$ generates the field. Let $A=\sum_{i=0}^{m-1}a_ix^i$ and $B=\sum_{i=0}^{m-1}b_ix^i$. Then, the product of A and B is computed as Algorithm 1.

Algorithm 1 The conventional multiplication algorithm over $GF(2^m)$

Require: $A,B,F(x)$
Ensure: $C=AB \mod F(x)$
1: $T_0=0$
2: for $i=1$ to m do begin
$T_i=T_{i-1}x+b_{m-i}A \mod F(x)$
end
3: return T_m.

3.2. Proposed CED multiplication algorithm

In this section, we will present an algorithm for CED multiplication using linear codes. We can enhance the calculation efficiency of Algorithm 1 by improving the parallelism degree. Let $A'=Ag(x)$ and $f(x)=F(x)g(x)$ where $F(x)$ and $g(x)$ are polynomials with degree m and n, respectively. After choosing a positive integer k as the parallel parameter, we present Algorithm 2 as follows. Here, let t_{end}, t_i be the vector form of $T_{m-(k-1)\lfloor m/k\rfloor}$, T_i, respectively; denote t_l and t_h as the vectors consisting of the first m coordinates and the remaining coordinates of t_{end}, respectively; G is the generator matrix according to $g(x)$ and G_1 and G_2 denote the first m columns and the remaining columns of matrix G, respectively.

In Algorithm 2, step 1 is used to obtain the product of A and $g(x)$. By iterating from step 2 to 4, the iteration result t_{end} which is the code word corresponding to C is obtained. Then, t_{end} is recovered to c in step 5. And step 6 is designed for error detection by checking whether t_{end} is a code word or not. Note that, step 6 can also be placed after T_i to check any intermediate result T_i. The effect of single and multiple checking points will be discussed in Section 5.

Algorithm 2 Proposed algorithm for CED multiplication using a linear code

Require: $A, B, F(x), g(x)$
Ensure: $C = AB \mod F(x)$
1: $A' = Ag(x)$
2: $T_0 = 0$
3: for $i = 1$ to $\lfloor m/k \rfloor$ do begin
$T_i = T_{i-1}x^k + \sum_{j=0}^{k-1} b_{m-j-(i-1)k-1} A' x^{k-j-1} \mod f(x)$
end
4: for $i = \lfloor m/k \rfloor + 1$ to $m-(k-1)\lfloor m/k \rfloor$ do begin
$T_i = T_{i-1}x + b_{m-i-\lfloor m/k \rfloor (k-1)} A' \mod f(x)$
end
5: get the result $T_{m-(k-1)\lfloor m/k \rfloor}/g(x)$ by $c = t_l G_1^{-1}$.
6: check the result by $t_h = t_l G_1^{-1} G_2$ or $Ht_{end}^T = 0$

4. Semi-systolic PB multiplier with CED capability

The proposed Algorithm 2 can be divided into the following four main processes as shown in Fig. 1:

1.Coding process (step 1)
2.Multiplying process (step 2-4)
3.Decoding process (step 5)
4.Checking process (step 6)

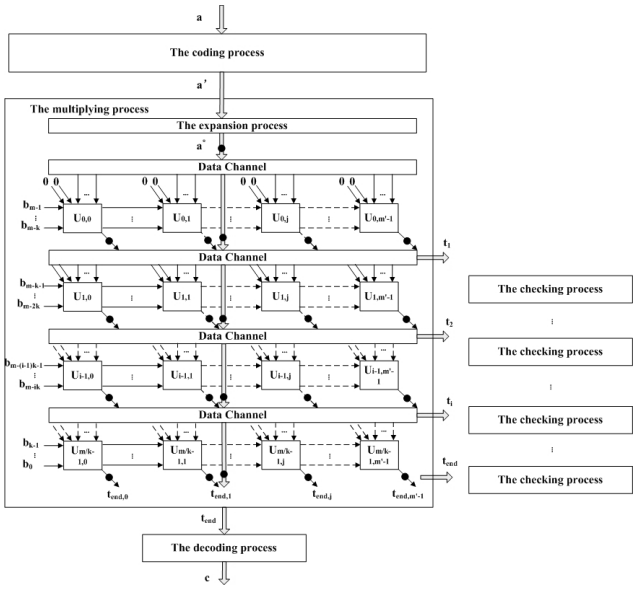


Figure 1 System framework for the multiplication

In the following, we assume that k divides m because the result is almost the same if k does not divide m. We will introduce the multiplying process in Section 4.1 and the other three processes in Section 4.2.

4.1. Implementation of multiplying process

Based on a choice of $F(x)$ and $g(x)$ according to Algorithm 2, we have

$$f(x) = F(x)g(x) = x^{m'} + \sum_{i=0}^{m'-1} f_i x^i$$

where $m' = m + n$ and m, n is the degree of $F(x), g(x)$, respectively. Then we will implement the multiplying process modulo $f(x)$.

Since $F(x)$ generates $GF(2^m)$, Let us denote

$$x^{m'+j} = \sum_{i=0}^{m'-1} f_i^{(j)} x^i \mod f(x)$$

$$A' = \sum_{i=0}^{m'-1} a'_i x^i$$

$$A^{(j)} = A' x^j = \sum_{i=0}^{m'-1} a_i^{(j)} x^i \mod f(x)$$

*Remark.*When k_0 is smaller compared with m, it is not difficult to find $F(x)$ and $g(x)$ satisfying the following conditions:

1.As the number of nonzero f_i, s is no more than 9.
2.$i - j \geq k_0, m' - i \geq k_0$ for any $f_i = 1, f_j = 1$.

Note that, k_0 in condition 2) determines the upper bound of the parallel parameter k in Algorithm 2, that is, $k_0 \geq k$.

When the two conditions in the remark hold, the expression for $f_i^{(j)}$ and $a_i^{(j)}$ can be simplified as follows

$$f_i^{(0)} = f_i \qquad for \;\; 0 \leq i \leq m' - 1 \tag{2}$$

$$f_i^{(j)} = \begin{cases} f_{i-j} & for \;\; j \leq i \leq m' - 1 \\ 0 & for \;\; 0 \leq i \leq j - 1 \end{cases} \tag{3}$$

where $j = 1, ..., k-1$.

$$a_i^{(0)} = a'_i \;\; for \;\; 0 \leq i \leq m' - 1$$

$$a_i^{(j)} = \begin{cases} a'_{i-j} + \sum_{t=1}^{j} a'_{m'-t} f_{i-j+t} \\ \qquad\qquad for \;\; j \leq i \leq m' - 1 \\ \sum_{t=j-i}^{j} a'_{m'-t} f_{i-j+t} \\ \qquad\qquad for \;\; 0 \leq i \leq j - 1 \end{cases} \tag{4}$$

where $j = 1, ..., k-1$.

Generally, we choose $F(x), g(x)$ for our CED multiplication, then $f(x)$ is pre-computed and fixed for constructing the semi-systolic array depicted in Fig. 1. It is implied that any expression determined by f_i can be considered to be known. According to the remark above, for one or none of f_{i-j+t} equals 1 in (4), there exists no more than $j \times s$ elements which do not equal to any of the coefficients of A', among $\{a_i^{(j)}, i = 0, 1, ..., m' - 1\}$. Thus,

for all $j \in \{1,2,...,k-1\}$, the number of such $a_i^{(j)}$s reaches to a total of $\frac{k(k-1)s}{2}$ and each one can be implemented by a 2-input XOR gate from the coefficients of A'. Then $m' + \frac{k(k-1)s}{2}$ bits, denoted as a^*, are required to be stored, which cover $\{a_i^{(j)}, i = 0,1,...,m'-1, j = 0,1,...,k-1\}$ in each row of the array. And each $a_i^{(j)}$ should be placed beside the jth column of the semi-systolic array for fast signal propagation. Moreover, we denote the transformation from a' which is the vector form of A' to a^* as expansion process which is a part of the multiplying process with a propagation delay T_X.

Based on the analysis above, step 3 in Algorithm 2 can be rewritten as

$$\begin{aligned} T_i &= T_{i-1}x^k + \sum_{l=0}^{k-1} b_{m-l-(i-1)k-1} A^{(k-l-1)} \mod f(x) \\ &= \sum_{l=k}^{m'-1} t_{i-1,l-k}x^l + \sum_{l=0}^{k-1} t_{i-1,l+m'-k}x^{l+m'} \\ &\quad + \sum_{l=0}^{k-1} b_{m-l-(i-1)k-1} A^{(k-l-1)} \mod f(x) \end{aligned}$$

where

$$t_{i,j} = \begin{cases} t_{i-1,j-k} + \sum_{l=0}^{k-1} b_{m-l-(i-1)k-1} a_j^{(k-l-1)} \\ \quad + \sum_{l=0}^{k-1} t_{i-1,l+m'-k} f_j^{(l)} \\ \qquad\qquad for \;\; j \geq k \\ \sum_{l=0}^{k-1} b_{m-l-(i-1)k-1} a_j^{(k-l-1)} \\ \quad + \sum_{l=0}^{k-1} t_{i-1,l+m'-k} f_j^{(l)} \\ \qquad\qquad for \;\; 0 \leq j < k \end{cases} \tag{5}$$

Based on condition 2) in the remark, $\sum_{l=0}^{k-1} t_{i-1,l+m'-k} f_j^{(l)}$ can be simplified into $t_{i-1,l_j+m'-k} f_{j-l_j}$ where $l_j \in \{0, 1,...,k-1\}$. Then (5) can be implemented by a basic cell which consists of $k+1$ 2-input AND gates and $k+1$ 2-input XOR gates as shown in Fig. 2. And the cell delay is $T_A + \lceil \log_2(k+2) \rceil T_X + T_L$. Moreover, the semi-systolic array for the multiplying process is built up by mm'/k basic cell in Fig. 1.

4.2. Implementation of coding, decoding and checking process

In this section, we consider the implementation of coding, decoding and checking processes based on the choice below.

$$\begin{aligned} F(x) &= F_3(x) \;\; or \;\; F_5(x) \\ g(x) &= x^{p_1} + 1 \\ h(x) &= \sum_{i=0}^{p_2-1} x^{ip_1} \\ x^{p_1p_2} + 1 &= g(x)h(x) \end{aligned}$$

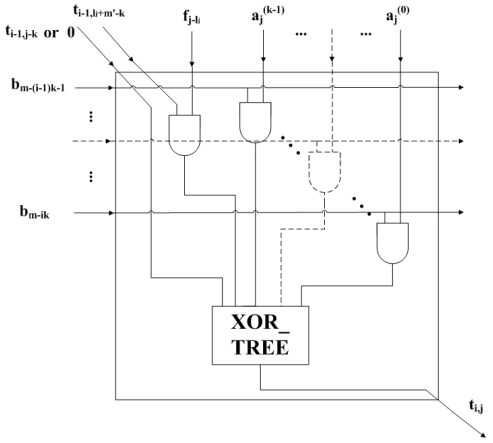


Figure 2 Basic cell $U_{i-1,j}$ for the multiplier

where F_3 and F_5 denote an irreducible trinomial and an irreducible pentanomial, respectively. Additionally, p_1 can be any positive integer and p_2 is determined by $p_1(p_2 - 1) \geq m > p_1(p_2-2)$. And $g(x)$ is a divisor of $x^{p_1p_2}+1$.

First, during the coding process of Algorithm 2, the relationship between A and A' can be expressed by $a' = aG$ where a and a' is the vector form of A and A', respectively. Then the coding process is implemented by the combinational logic. The area and time complexity of the coding rely on $g(x) = x^{p_1} + 1$. The critical path delay is T_X and the area consumption is $m - p_1$ 2-input XOR gates.

Second, the decoding process $c = tG_1^{-1}$ is realized by matrix multiplication Similarly, the critical path delay is $\lceil \log_2(1 + \lfloor \frac{m-1}{p_1} \rfloor) \rceil \;\; T_X$ and the area consumption is $m\lfloor \frac{m-1}{p_1} \rfloor - \frac{p_1}{2}\lfloor \frac{m-1}{p_1} \rfloor - \frac{p_1}{2}\lfloor \frac{m-1}{p_1} \rfloor^2$ 2-input XOR gates.

Third,complexity for the checking process is to be analyzed. According to Theorem 2, a cyclic $(p_1p_2, p_1p_2 - p_1)$ code C over $\mathbb{F}_2$ can be obtained by multiplying each message (identified with a polynomial of degree $m_c < p_1p_2 - p_1$) by a fixed polynomial $g(x) = x^{p_1} + 1$ which is a divisor of $x^{p_1p_2} + 1$. The generator and parity-check matrix is G and H determined by $g(x)$ and $h(x)$, respectively, where $h(x) = (x^{p_1p_2} - 1)/g(x) = \sum_{i=0}^{p_2-1} x^{ip_1}$. By multiplying each message (identified with a polynomial of degree $m \leq m_c$) by the same polynomial $g(x)$, a linear $(m + p_1, m)$ code L over $\mathbb{F}_2$ can be obtained according to Theorem 1. Then the parity-check matrix H' is actually a submatrix of H consisting of the first $m + p_1$ columns. Similarly, the generator matrix G' is the submatrix of G consisting of the first m rows and the first $m + p_1$ columns. The inequality $p_1(p_2 - 1) \geq m > p_1(p_2 - 2)$ ensures that H and G are the minimum matrixs as described above. Thus the area complexity of checking process depends on the weight of H' and the time

complexity depends on the maximum weight of each row in H'. Then the critical path delay is $\lceil \log_2 p_2 \rceil\ T_X$ and the area consumption is $mp_2 - m - p_1 p_2^2 + 3p_1 p_2 - 2p_1 + \sum_{i=0}^{p_1 p_2 - p_1 - m - 1}(p_2 - 1 - \lceil \frac{i}{p_1} \rceil)$ 2-input XOR gates.

In addition, we can divide the critical path of the coding, decoding and checking processes into several short paths whose delays are slightly less than the system clock using pipeline technology. In this way, the throughput of the three processes can be increased to one result per clock in accordance with the multiplying process.

5. Error detection capability for all-cell error model

In Section 4, a concurrent error detection semi-systolic PB multiplier based on coding theory has been introduced. In this section, we create a new all-cell error model for systolic or semi-systolic multipliers which do not exist in the literature. Error detection capability is presented based on the new model, additionally with the extra time and area overheads.

5.1. All-cell error model

The error model will be established according to the systolic or semi-systolic array architecture of multipliers. In previous papers, analysis of error detection is mostly based on the the condition that single stuck-at or single-cell errors would occur in systolic or semi-systolic arrays. However, transient or multiple-cell errors are more likely to occur in practice which render the CED capability invalid to some sense. For accurate result, we do not restrict any characteristics of errors in our error model. In other words, we do not care whether errors locate in a single cell or multiple cells., whether the error is transient or permanent, and whether the error is natural or manmade and so on.

For the clarity of analysis, we assume that k divides m. Let E_i denote the error occurs in the ith row with the vector form $e_i = (e_{i,0}, e_{i,1}, ..., e_{i,m'-1})$. And $e_i = (0,0,...,0)$ means error does not exit. Reviewing Algorithm 2, step 3 shows the data relation between two adjacent rows of the semi-systolic array in Fig. 1. After embedding error vectors into each row, step 3 is transformed into

$$T_i = T_{i-1}x^k + \sum_{j=0}^{k-1} b_{m-j-(i-1)k-1} A' x^{k-j-1} + E_{i-1} \bmod f(x)$$

So based on the premise that $g(x)$ divides T_{i-1}, we can easily detect nonzero E_{i-1} if T_i is not divisible by $g(x)$. The error nonzero E_{i-1} can not be detected if $g(x)$ divides E_{i-1}, for the reason that $g(x)$ divides T_i if and only if $g(x)$ divides E_{i-1}. Then we will compute the probability of error detection of nonzero E_{i-1} called P_{D1} and the probability of undetected E_{i-1} called P_{U1}. Let p be the probability of $e_{ij} = 1$, which means a mistake output bit for $t_{i,j}$ occurring in the basic cell $U_{i-1,j}$. Let W_j be the number of undetected E_i with the hamming weight j. Then the probability of an undetected nonzero E_i is

$$P_{U1} = \sum_{i=1}^{m'} W_i p^i (1-p)^{m'-i} \tag{6}$$

Obviously, $P_{D1} = 1 - P_{U1}$. If the checking process described in step 6 of Algorithm 2 is placed after each row, the total probability of error detection is $1 - P_{U1}^{\frac{m}{k}}$. However, the area complexity may be unacceptable. Then we consider placing a checking process after every i_{fix} rows in the array. Assuming that $g(x)$ divides T_{i-1}, influenced by $E_{i-1}, ..., E_{i+j-1}$ which are independent identically distributed, the correct T_{i+j} is converted to

$$T'_{i+j} = T_{i+j} + \sum_{v=0}^{j} E_{i+v-1} x^{k(j-v)} \bmod f(x) \tag{7}$$

Thus under the condition that $g(x)$ divides T_{i-1}, we can detect the existence of $E_{i-1}, ..., E_{i+j-1}$, which are not all zero, when $g(x)$ does not divide T'_{i+j}. The errors $E_{i-1}, ..., E_{i+j-1}$ can not be detected when $g(x)$ exactly divides $\sum_{v=0}^{j} E_{i+v-1} \cdot x^{k(j-v)}$.

5.2. Error detection capability

Before giving out the error detection capability for all-cell error model, we will limit the probability of multiple undetected error by an upper bound. We firstly introduce the following theorem to determine the upper bound.

Theorem 3. *Suppose that $p_{i,j}$ is the probability that $\sum_{v=0}^{i} E_v x^{k(i-v)} \equiv J \mod g(x)$ where $i = 0,1,2,3,...$ and j is the vector form of the polynomial J. Then, $\lim_{i\to\infty} p_{i,j} = \frac{1}{2^n}$ holds.*

Proof. From (6), it is clear that

1. $p_{i,j} \neq 1$ and $p_{i,j} \neq 0$;
2. $\sum_j p_{i,j} = 1$;
3. $p_{i,j} = \Sigma_v p_{i-1,v} p_{0,x^k(v)\oplus j}$.

where v and $x^k(v)$ is the vector form of the polynomial V and $x^k \cdot V$, respectively.

Let $m_i = \max_{j,l}\{(p_{i,j} - p_{i,l})\}$

$$\begin{aligned} m_{i+1} &= \max_{j,l}\{(p_{i+1,j} - p_{i+1,l})\} \\ &= \max_{j,l}\{(\Sigma_v p_{i,v} p_{0,x^k(v)\oplus j} - \Sigma_v p_{i,v} p_{0,x^k(v)\oplus l})\} \\ &= \max_{j',l'}\{\Sigma_s p_{0,s}(p_{i,j'} - p_{i,l'})\} \\ &\leq \Sigma_s p_{0,s} \max_{j',l'} |p_{i,j'} - p_{i,l'}| \\ &\leq m_i \end{aligned} \tag{8}$$

where

$$(j',s) \ \ satisfies \ \ s = x^k(j') \oplus j$$
$$(l',s) \ \ satisfies \ \ s = x^k(l') \oplus l$$

So the series $\{m_i\}$ is monotone decreasing and it has the lower bound of 0, implying that $\{m_i\}$ has a limit M, i.e., $\forall \varepsilon > 0, \exists N \in \mathbb{N}$, such that, for $\forall i > N$

$$M < m_i < M + \varepsilon,$$
$$M < m_{i+1} < M + \varepsilon,$$

Therefore, $m_i - m_{i+1} < \varepsilon$.

Suppose that the limit $M > 0$. Easy to know, there exists (j,l) such that $p_{i,j'} - p_{i,l'} \leq 0$. Then from (8) we have

$$\begin{aligned} m_i - m_{i+1} &= m_i - \max_{j',l'}\{\Sigma_s p_{0,s}(p_{i,j'} - p_{i,l'})\} \\ &= \min_{j',l'}\{\Sigma_s p_{0,s}(m_i - (p_{i,j'} - p_{i,l'}))\} \\ &\geq m_i \min_s(p_{0,s}) \\ &\geq M \min_s(p_{0,s}) \end{aligned}$$

It is obvious that the formula is contradictory with $m_i - m_{i+1} < \varepsilon$, contradicting the suppose about $M > 0$. So $M = 0$, that is, $\{m_i\}$ has a limit 0.

Therefore, $\lim_{i\to\infty} p_{i,j} = \frac{1}{2^n}$ is proven.

Theorem 3 implies that the probability of undetected errors of $E_0, E_1, ..., E_{i-1}$ in (7) is

$$P_{Ui} = p_{i,0} - (1-p)^{im'}$$

So P_{Ui} infinitely approaches to $\frac{1}{2^n}$ when i is large enough. According to definition of limit, $\forall \varepsilon > 0, \exists N \in \mathbb{N}$, such that $\frac{1}{2^n} - \varepsilon < P_{Ui} < \frac{1}{2^n} + \varepsilon$ for $\forall i > N$. Then $i_{fix} > N$ is fixed which implies that the probability of detecting the errors of i_{fix} rows has a lower bound of $(1 - \frac{1}{2^n} - \varepsilon)$. Moreover, the probability of all-cell error detection for our multiplier is at least $1 - (\frac{1}{2^n} + \varepsilon)^{\lfloor \frac{m}{ki_{fix}} \rfloor}$.

Then we obtain the relationship between n and extra overheads of time and area for realizing the CED capability under the NIST recommended degree $m = 571$, where n is the degree of $g(x)$. As is shown in Fig. 3, the extra time overhead declines with the growth of n and is always kept below 1.1%, and the extra area overhead percentage has a minimum at $n = 8$ when different number of checking processes are inserted into the multiplier. In other words, error detection efficiency is relatively high when $n = 8$. In addition, Table 1 shows that only 4 checking points are required for reaching to all-cell error detection probability 99.999999% and the extra time and area overhead is 0.89% and 3.160% when $k = 2, n = 8$, respectively.

6. Comparisons

In the CMOS VLSI technology, transistor count is 6, 6, 6, 6, 16, 8 and latency is 7, 8, 12, 4, 16, 13ns respectively for 2-input AND, 2-input OR, 2-input XOR, 2-1 multiplexer, 4-1 multiplexer and 1-bit latch, respectively. Considering the multiplying process as a multiplier without CED capability, we compare its time complexity, area complexity and time-area product with some other bit-parallel systolic multipliers. Since the irreducible polynomial generating the finite field can be trinomials or pentanomials with the degree less than 10,000, let $m = 571$, $s = 4$, $k = 6$, $n = 8$, $d = 5$, $t = 2$ in the following comparisons whose results are shown in Table 2. Comparison has been done between the proposed multiplier and systolic or semi-systolic CED multipliers in [17], [18] and [16] which is constructed based on DB, NB and PB, respectively. In the comparison with [18] and [16], the proposed multiplier saves 70% and 74% time overhead, and 70% and 63% area overhead, respectively.

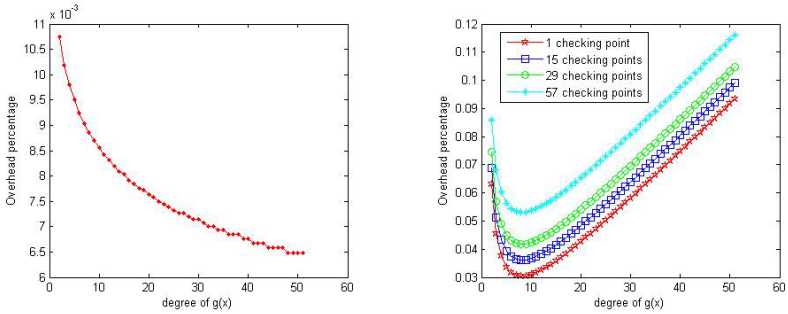
Furthermore, multipliers [18], [16] have constructed a model based on single stuck-at or single-cell errors and their error detection probability is 100%. However, transient or multiple-cell errors are likely to occur in practice. In this paper, we have created an all-cell error model for systolic and semi-systolic finite field multipliers. Remarkably, our error detection probability is 99.999999% with low extra time and area overheads. And the error detection probability can be adjusted by changing the number of checking processes. Moreover, our multiplier has a throughput of 1 per cycle which is lager than others.

7. Conclusion

In this paper, a concurrent all-cell error detection semi-systolic PB multiplier based on coding theory is proposed. Application range of our multiplier covers almost all finite fields including the NIST recommended five ones. Additionally, an all-cell error model is creatively built for systolic or semi-systolic finite field multipliers. The proposed multiplier efficiently reduces time and area complexity. Specifically, our multiplier with CED capability saves at least 70% time complexity and 63% area complexity compared with some other systolic or semi-systolic CED multipliers. Significantly, based on the new error model, the error detection capability is analyzed and the probability is strictly proven to be 99.999999% with 2.088% extra time overhead and 4.978% extra area overhead for CED, being more accurate than simulation results.

Acknowledgement

The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.



(*a*) extra time overhead percentage (*b*) extra area overhead percentage

Figure 3 Time and area overhead fixing $m = 571$

Table 1 Relationship among number of checking points, error detection probability and area overhead

Number of checking points	1	2	3	4	...
Error detection probability	99.9609375%	99.998474%	99.999994%	99.999999%	...
Area overhead	3.039%	3.079%	3.120%	3.160%	...
Time overhead	0.89%				

Table 2 Comparison of Various CED Systolic or Semi-Systolic Multipliers over $GF(2^m)$

Multiplier	C. W. Chiou[18]	W. T. Huang[16]	The proposed multiplier (Fig. 1)
Basis	normal	polynomial	pre-computed polynomial
Error type	single-cell	single-cell	all-cell
Cell delay	$T_A + T_X + T_L$	E:$T_{2M} + T_A + T_X + T_L$ V:$T_{3A} + T_X + T_L$	CE:$2T_X + T_L$ MU:$T_A + \lceil \log_2(k+1) \rceil T_X + T_L$ DE:$\lceil \log_2(m/n+1) \rceil T_X$ CH:$\lceil \log_2(m/n+1) \rceil T_X$
Latency	$mt/2+1$	$m+2$	$m/k+2$
Total delay	$16mt+32$	$36m+71$	$12\lceil \log_2(m/n+1) \rceil + 28m/3 + 37$
Transistor counts	$15(mt)^2+28mt+22$	$48(m+1)^2+100m+50$	$209m^2/12+979m/3+905$
Extra time	0.55%	12.9%	2.088%
Extra area	0.23%	0.72%	4.978%
Throughput	1/2	1/4	1
ECD probability	100%	100%	99.999999%

Notes:
2)CE denotes the coding process and the expanding process
3)MU denotes basic cell of the multiplying process
4)DE, CH denote the decoding and the checking process, respectively

References

[1] R. Lidl, H. Niederreiter, Introduction to Finite Fields and Their Applications, Cambridge Univ., New York, 1994.

[2] C.Y. Lee, E.H. Lu, and J.Y. Lee, Bit-parallel systolic multipliers for $GF(2^m)$ fields defined by all-one and equally spaced polynomials, IEEE Trans. Computers **50**, 385-393 (2001).

[3] H. Wu, M.A. Hasan, and I.F. Blake, New low-complexity bit-parallel finite field multipliers using weakly dual bases, IEEE Trans. Computers **47**, 1223-1234 (1998).

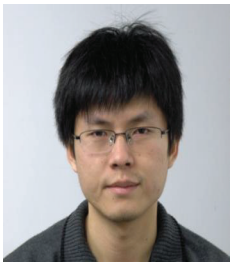
[4] A. Reyhani-Masoleh, Efficient algorithms and architectures for field multiplication using gaussian normal bases, IEEE Trans. Computers **55**, 34-47 (2006).

[5] S. Fenn, M. Gossel, M. Benaissa, and D. Taylor, On-line error detection for bit-serial multipliers in $GF(2^m)$, J.Electronic Testing: Theory and Applications **13**, 29-40 (1998).

[6] A. Reyhani-Masoleh and M.A. Hasan, Error detection in polynomial basis multipliers over binary extension fields, Cryptographic Hardware and Embedded Systems - Ches 2002, Lecture Notes in Computer Science **2523**, 515-528 (2002).

[7] A. Reyhani-Masoleh and M. Hasan, Fault detection architectures for field multiplication using polynomial bases,

IEEE Trans. Computers **55**, 1089-1103 (2006).
[8] A. Hariri and A. Reyhani-Masoleh, Fault detection structures for the montgomery multiplication over binary extension fields, Proc. 4th International Workshop on Fault Diagnosis and Tolerance in Cryptography, 37-43 (2007).
[9] A. Hariri and A. Reyhani-Masoleh, Concurrent error detection in montgomery multiplication over binary extension fields, IEEE Trans. Computers **60**, 1341-1353 (2011).
[10] S. Bayat-Sarmadi and M.A. Hasan, Concurrent error detection of polynomial basis multiplication over extension fields using a multiple-bit parity scheme, Proc. 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 102-110 (2005).
[11] W. Chelton and M. Benaissa, Concurrent error detection in $GF(2^m)$ multiplication and its application in elliptic curve cryptography, IET Circuits, Devices and Systems **2**, 289-297 (2008).
[12] C.W. Chiou, W.Y. Liang, and H.W. Chang et al., Concurrent error detection in semi-systolic dual basis multiplier over $GF(2^m)$ using self-checking alternating logic, Iet Circuits Devices and Systems **4**, 382-391 (2010).
[13] S. Bayat-Sarmadi and M.A. Hasan, Concurrent error detection in finite-field arithmetic operations using pipelined and systolic architectures, IEEE Trans. Computers **58**, 1553-1567 (2009).
[14] C.W. Chiou, Concurrent error detection in array multipliers for $GF(2^m)$ fields, Electronics Letters **38**, 688-689 (2002).
[15] C.Y. Lee, C.W. Chiou, and J.M. Lin, Concurrent error detection in a polynomial basis multiplier over $GF(2^m)$, Journal of Electronic Testing-Theory and Applications **22**, 143-150 (2006).
[16] W.T. Huang, C.H. Chang, and C.W. Chiou et al., Concurrent error detection and correction in a polynomial basis multiplier over $GF(2^m)$, Iet Information Security **4**, 111-124 (2010).
[17] C.W. Chiou, C.Y. Lee, and J.M. Lin et al., Concurrent error detection and correction in dual basis multiplier over $GF(2^m)$, Iet Circuits Devices and Systems **3**, 22-40 (2009).
[18] C.W. Chiou, C.C. Chang, and C.Y. Lee et al., Concurrent error detection and correction in gaussian normal basis multiplier over $GF(2^m)$, IEEE Trans. Computers **58**, 851-857 (2009).
[19] S.Bayat-Saramdi and M.A. Hasan, Run-time error detection in polynomial basis multiplication using linear codes, IEEE Intern. Conf. Application-specific Systems, Architectures and Processors(ASAP-2007), 204-209 (2007).
[20] G. Gaubatz and B. Sunar, Robust finite field arithmetic for fault-tolerant public-key cryptography, Proc. 3rd International Workshop on Fault Diagnosis and Tolerance in Cryptography **4236**, 196-210 (2006).
[21] Arash Reyhani-Masoleh and M. Anwar Hasan, Towards fault tolerant cryptographic computations over finite fields, ACM Transactions on Embedded Computing Systems **3**, 593-613 (2004).
[22] Bijan Ansari, Ingrid Verbauwhede, A hybrid scheme for concurrent error detection of multiplication over finite fields, 2010 IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems, 399-407 (2010).
[23] R. Katti, and J. Brennan, Low complexity multiplication in a finite field using ring representation, IEEE Trans. Computers **52**, 418-427 (2003).



Wangjie Qiu was born in Jiangsu, China in 1986. He received his Bachelor of Science degree in Information and Computer Science from Beihang University, Beijing, China in 2008. He began his Ph. D. research in the area of Cryptography, Coding and their Hardware Implementation in 2008.