Journal of Information Technology and Engineering
Vol. 3 No. 2 (December, 2018)

Efficient Systolic Arrays for Power-Sum, Inversion, and Division in GF(2^m)

Che-Wun CHIOU[1], Chiou-Yng LEE[2], Jim-Min LIN[3]

[1]Dept. of Computer Science & Information Engineering, Ching Yun University
Chung-Li 320, Taiwan, R.O.C., *E-mail: cwchiou@cyu.edu.tw*

[2]Dept. of Computer Information and Network Engineering, LungHwa University of Science and Technology
Taoyuan County 333, Taiwan, R.O.C., *E-mail: PP010@mail.lhu.edu.tw*

[3]Dept. of Information Engineering and Computer Science, Feng Chia University
Taichung City 407, Taiwan, R.O.C.*E-mail: jimmy@fcu.edu.tw*

Received: 13th August 2017 Revised: 24th December 2017 Accepted: 10th March 2018

Abstract: *Based on Fermat's theorem and polynomial basis, an efficient systolic array for power-sum in GF(2^m) with the circuit folding technique is presented. A power-sum algorithm based on the reuse of two-folded systolic array circuit for '*AB^2+C*', is firstly proposed in this article. The proposed systolic power-sum architecture saves half of space complexity as compared to other existing semi-systolic power-sum circuits. Also in this paper, the circuits of two important computations: inversion and division, which are based on the proposed power-sum circuit, are then presented. Both proposed circuits also save 75% of space complexity and 50% of time complexity while comparing with other off the shelf inversion/division circuits which employ the Fermat algorithm.*

Key words: *Galois Field, Polynomial Basis, Cryptography, Power-Sum, Inversion, Division, Fermat Theorem.*

1. INTRODUCTION

Finite field arithmetic operations have several applications in coding theory [1], cryptography [2], digital signal processing [3-4], switching theory [5], and pseudorandom number generation [6], and so on. Arithmetic operations in such areas require several complex operations, like multiplication, power-sum ($C+AB^2$), inversion/division, and exponentiation. The power-sum operation is a basic operation for public-key cryptosystem [7] such as RSA [8] and Elliptic curve cryptosystem [9] and in decoding multiple-error-correcting binary BCH codes and RS codes [10-12]. Numerous researchers proposed many efficient power-sum architectures [13-18]. Most power-sum architectures are based on the polynomial basis representation of GF(2^m). However, the major shortcomings of such circuits, as regards cryptographic applications, are their high space and time complexities. Thus, further research on efficient power-sum architectures with low space and time complexities is elegantly needed. In this paper, a systolic array implementation of the power-sum circuit with low space complexity by employing the circuit folding technique is proposed.

Inversions and divisions are essential operations in many error-control coding schemes for reliable data transmission and storage systems, and for many cryptographic applications such as Diffe-Hellman key exchange algorithm [7], RSA algorithm [8], elliptic curve cryptography [9], and elliptic curve digital signature algorithm [9,19]. Three well-known methods for finding an inverse element in a finite field are the table lookup algorithm, the extended Euclid's algorithm [20], and the repeated exponentiation algorithm [21]. The former two algorithms are not easily realized in a VLSI circuit. The exponentiation algorithm based on the Fermat's theorem [22] uses the iterative multiply-square algorithm. Such multiply-square algorithm can be realized by the power-sum operation. The Fermat's theorem is employed in this paper.

The performance of finite field arithmetic operations is highly related to the representation of the field elements. There are three main popular types of bases over finite fields, namely polynomial basis (PB), normal basis (NB), and dual basis (DB). The polynomial basis representation [23-33] is widely used and leads to efficient implementations of finite field arithmetic operations. As compared to other two bases representations, the polynomial basis representation has the features of low design complexity. Additionally, PB architecture has three significant features, simplicity, regularity, and modularity. Therefore, it could be potentially fit to various applications. Regarding the normal basis representation [34-41], one important advantage is that the squaring of an element is computed by a cyclic shift of the binary representation. The dual basis representation [42-44] while requires less chip area than other two basis representations. This study is relied on the polynomial basis representation.

In this article, a new systolic array architecture employing the circuit folding technique for performing power-sum operation is presented to achieve the goal of saving space complexity and retaining same time complexity. Applying the proposed power-sum circuit, the inversion/division circuits utilizing Fermat's theorem are then presented. Such inversion/division circuits also have the features of low circuit complexity and short latency.

The remainder of this article is organized as follows. Section 2 briefly reviews the mathematical background. Section 3 presents the proposed power-sum circuit by utilizing the circuit folding technique. In Section 4, we present the inversion circuit based on the proposed power-sum circuit. The new division circuit will then be discussed in Section 5. A brief conclusion is made in the final section, Section 6.

2. MATHEMATICAL BACKGROUND

It is assumed that the reader is familiar with the basic concepts of finite fields. For more information, the reader can refer to [2]. In the following paragraphs, the results from the finite fields are briefly reviewed.

Let GF(2^m) be a finite field of 2^m elements. GF(2^m) is an extension field of the ground field GF(2). Let ± be a root of an irreducible polynomial of degree m over GF(2) given as $P(x) = p_0 + p_1x^1 + p_2x^2 + ... + p_{m-1}x^{m-1} + x^m$ where $p_0 = 1$.

Thus, the set $\psi = \{1, \alpha, \alpha^2, \alpha^3, ..., \alpha^{m-1}\}$ is a polynomial basis of GF(2^m). Any elements *A, B, C, Y*ÎGF(2^m) can be represented by

$$\begin{aligned} A &= a_0 + a_1\alpha + a_2\alpha^2 + ... + a_{m-1}\alpha^{m-1}, \\ B &= b_0 + b_1\alpha + b_2\alpha^2 + ... + b_{m-1}\alpha^{m-1}, \\ C &= c_0 + c_1\alpha + c_2\alpha^2 + ... + c_{m-1}\alpha^{m-1}, \\ Y &= y_0 + y_1\alpha + y_2\alpha^2 + ... + y_{m-1}\alpha^{m-1}, \end{aligned}$$

where $a_i, b_i, c_i, y_i \in GF(2)$ for all $0 \le i \le m-1$.
Let $Y=C+AB^2 \bmod P(x)$, the result is given by

$$\begin{aligned} Y &= C + AB^2 \bmod P(x) \\ &= C + A\times\left(b_0 + b_1\alpha + b_2\alpha^2 + ... + b_{m-1}\alpha^{m-1}\right)^2 \\ &= C + A\times\left(b_0 + b_1\alpha^2 + b_2\alpha^4 + ... + b_{m-1}\alpha^{2(m-1)}\right) \end{aligned} \quad (1)$$

From Fermat's theorem, for every $B \in GF(2^m), B^{2^m} = B$ and therefore we have

$$\begin{aligned} B^{-1} &= B^{2^m-2} \\ &= B^{2+2^2+2^3+...+2^{m-1}} \\ &= B^2B^{2^2}B^{2^3}...B^{2^{m-1}} \\ &= B^2\times(B^2)^2\times((B^2)^2)^2...\times\overbrace{((...((B)^2)^2)^2...)^2}^{m-1}. \end{aligned} \quad (2)$$

Based on Eq. (2), the inversion can be performed by repeating multiply and square algorithms. In other words, the inversion operation can be done iteratively by power-sum operations.

The division operation A/B is equivalent to the multiplication operation $A\times B^{-1}$, and is thus expressed as follows:

$$\begin{aligned} A/B &= A\times B^{-1} \\ &= A\times B^2\times(B^2)^2\times((B^2)^2)^2...\times\overbrace{((...((B)^2)^2)^2...)^2}^{m-1} \end{aligned} \quad (3)$$

Observing Eq. (3), the division operation can also be done by power-sum operations.

In this study, m is assumed to be an even number. If m is not an even number, m+1 is temporarily used by adding an extra 0 to the most significant bit and the computing result is then modified for the final correct result.

3. THE PROPOSED POWER-SUM OPERATION IN GF(2^M)

The power-sum computations are always required in decoding BCH codes and RS codes, computing inversions, and computing divisions. Using polynomial basis representation, Wei [14] presented a systolic power-sum circuit with bidirectional data flow. However, such a systolic array with bidirectional data flow is not suited to testable design. For gaining advantages of low space complexity, short latency, and fault tolerance, Wang and Guo [15] also employed polynomial basis to present a systolic array for power-sum computation with unidirectional data flow. Instead of the LSB-first schemes in conventional power-sum circuits, Kim *et al.* [16] used the MSB-first scheme to further reduce the space and time complexities in existing power-sum circuits. However, such existing systolic power-sum architectures still have shortcomings of high space complexity and long latency as such power-sum circuits are applied to cryptographic application. Wei [17] provided a semi-systolic design of the power-sum circuit for maximum throughput rate. Lee *et al.* [18] developed a time-independent bit-parallel systolic architecture for further saving space complexity. In this study, a new systolic power-sum array architecture by employing the circuit folding technique is presented for lower space complexity as compared to other existing power-sum circuits.

Assuming that the squaring of the element B is split into two sub-words,

$$B^2 = B_L + x^m B_H$$

where

$$B_L = b_0 + b_1\alpha^2 + b_2\alpha^4 + \ldots + b_{\frac{m}{2}-1}\alpha^{m-2}$$

$$B_H = b_{\frac{m}{2}} + b_{\frac{m}{2}+1}\alpha^2 + \ldots + b_{m-1}\alpha^{m-2}.$$

With the circuit folding technique, the power-sum equation in Eq.(1) can be rewritten as

$$\begin{aligned} Y &= C + AB^2 \bmod P(x) \\ &= C + A\times\begin{pmatrix}\left(b_0 + b_1\alpha^2 + b_2\alpha^4 + \ldots + b_{\frac{m}{2}-1}\alpha^{m-2}\right) + \\ \alpha^m\left(b_{\frac{m}{2}} + b_{\frac{m}{2}+1}\alpha^2 + \ldots + b_{m-1}\alpha^{m-2}\right)\end{pmatrix} \\ &= C + A\times\begin{pmatrix}\left(b_0 + b_1\alpha^2 + b_2\alpha^4 + \ldots + b_{\frac{m}{2}-1}\alpha^{m-2}\right) + \\ \alpha^m B_H\end{pmatrix} \\ &= \left(c_0 + c_1\alpha + c_2\alpha^2 + \ldots + c_{m-1}\alpha^{m-1}\right) + \left(\left(\left(\left(\left(AB_H\right)\alpha^2 + Ab_{\frac{m}{2}-1}\right)\alpha^2 + Ab_{\frac{m}{2}-2}\right)\alpha^2 + \ldots\right)\alpha^2 + Ab_0\right) \end{aligned} \tag{4}$$

$$\begin{aligned} F &= AB_H \\ &= A\times\left(b_{\frac{m}{2}} + b_{\frac{m}{2}+1}\alpha^2 + b_{\frac{m}{2}+2}\alpha^4 + \ldots + b_{m-1}\alpha^{m-2}\right) \\ &= \left(\left(\left((0)\alpha^2 + Ab_{m-1}\right)\alpha^2 + Ab_{m-2}\right)\alpha^2 + \ldots\right)\alpha^2 + Ab_{\frac{m}{2}} \end{aligned} \tag{5}$$

Before computing Eq.(4), the Eq.(5) must be performed in prior. The result of Eq.(5) is as the initial value of Eq.(4). Both Eq.(4) and Eq.(5) have the same iterative form and such iterative form is shown as follows:

$$T_{i+1} = T_i \times \alpha^2 + Ab_k \tag{6}$$

where

$$T_0 = \begin{cases} AB_H & \text{for Eq.(4)} \\ 0 & \text{for Eq.(5)} \end{cases}$$

$$k = \begin{cases} \frac{m}{2} - i - 1 & \text{for Eq.(4)} \\ m - i - 1 & \text{for Eq.(5)} \end{cases}$$

and

$$T_j \in GF(2^m) \text{ for } 0 \le j \le \frac{m}{2}$$

Suppose F and T_i are expressed as

$$F = f_0 + f_1\alpha + f_2\alpha^2 + \ldots + f_{m-1}\alpha^{m-1} \text{ and}$$

$$T_i = t_{i,0} + t_{i,1}\alpha + t_{i,2}\alpha^2 + \ldots + t_{i,m-1}\alpha^{m-1}$$

where

$$f_i, t_{i,j} \in GF(2), 0 \le i, j \le m\text{-}1.$$

Because α is the root of $P(x)$, thus $P(\alpha) = 0$ and we have the following results

$$\alpha^m = p_0 + p_1\alpha + p_2\alpha^2 + \ldots + p_{m-1}\alpha^{m-1} \tag{7}$$

$$\alpha^{m+1} = p_0' + p_1'\alpha + p_2'\alpha^2 + \ldots + p_{m-1}'\alpha^{m-1} \tag{8}$$

where

$$p_i' = p_{m-1}p_i + p_{i-1} \text{ for } 1 \le \text{i} \le m\text{-}1 \text{ and}$$

$$p_0' = p_{m-1}p_0$$

Substituting Eqs.(7-8) into Eq.(6), we obtain

$$\begin{aligned} T_{i+1} &= T_i \times \alpha^2 + Ab_k \\ &= \begin{pmatrix} \left(b_k a_0 + t_{i,m-2}p_0 + t_{i,m-1}p_0'\right) + \left(b_k a_1 + t_{i,m-2}p_1 + t_{i,m-1}p_1'\right)\alpha + \\ \left(b_k a_2 + t_{i,m-2}p_2 + t_{i,m-1}p_2' + t_{i,0}\right)\alpha^2 + \left(b_k a_3 + t_{i,m-2}p_3 + t_{i,m-1}p_3' + t_{i,1}\right)\alpha^3 + \\ \ldots + \left(b_k a_{m-1} + t_{i,m-2}p_{m-1} + t_{i,m-1}p_{m-1}' + t_{i,m-3}\right)\alpha^{m-1} \end{pmatrix} \end{aligned} \tag{9}$$

Based on the Eq. (9), the semi-systolic array architecture with $\frac{m}{2}\times m$ cells for Eq. (4) is shown in Fig.1. The detailed circuit of the cell U in Fig.1 is depicted in Fig.2. The symbol D^i in Fig.1 denotes i clock cycles delay. The function unit L in Fig.2 is a 1-bit latch. At the first round, the systolic array in Fig.1 is used for computing $F = A \times B_H$. At the second round, the result F obtained at the first round is applied as input to compute the final result Y. The inputs applied to such a systolic array are shown in Fig.1.

As comparing with other existing power-sum array architectures, the following assumptions for space complexity are made: (1) a 2-input AND gate, a 1-bit latch, a 2-input XOR gate, 2-to-1 MUX, 2×1 Switch, and 3×1 Switch consist of 6, 8, 6, 6, 6, and 11 transistors, respectively [45]; (2) an 3-input XOR gate and an 4-input XOR gate are constructed by 2 2-input XOR gates and 3 2-input XOR gates, respectively. The comparison results of various power-sum array architectures are depicted in Table 1. Our proposed systolic array architecture for the power-sum circuit saves about 50% and 15% of space complexity to existing power-sum array architecture in [17] and [18], respectively. Moreover, our proposed power-sum array architecture only requires one cell type, but the array architecture in [18] needs three types of cells.

The propagation delays through one cell of two different array architectures in Table 1 are assumed the same because the propagation delays for both 3-input XOR gate and 4-input XOR gate have the same propagation delay. Table 1 shows that our proposed power-sum array architecture is executed as fast as Wei's power-sum array architecture [17]. Furthermore, the data flow of our proposed power-sum array architecture is unidirectional. Unidirectional data flow makes fault-tolerant circuit design easy and feasible.

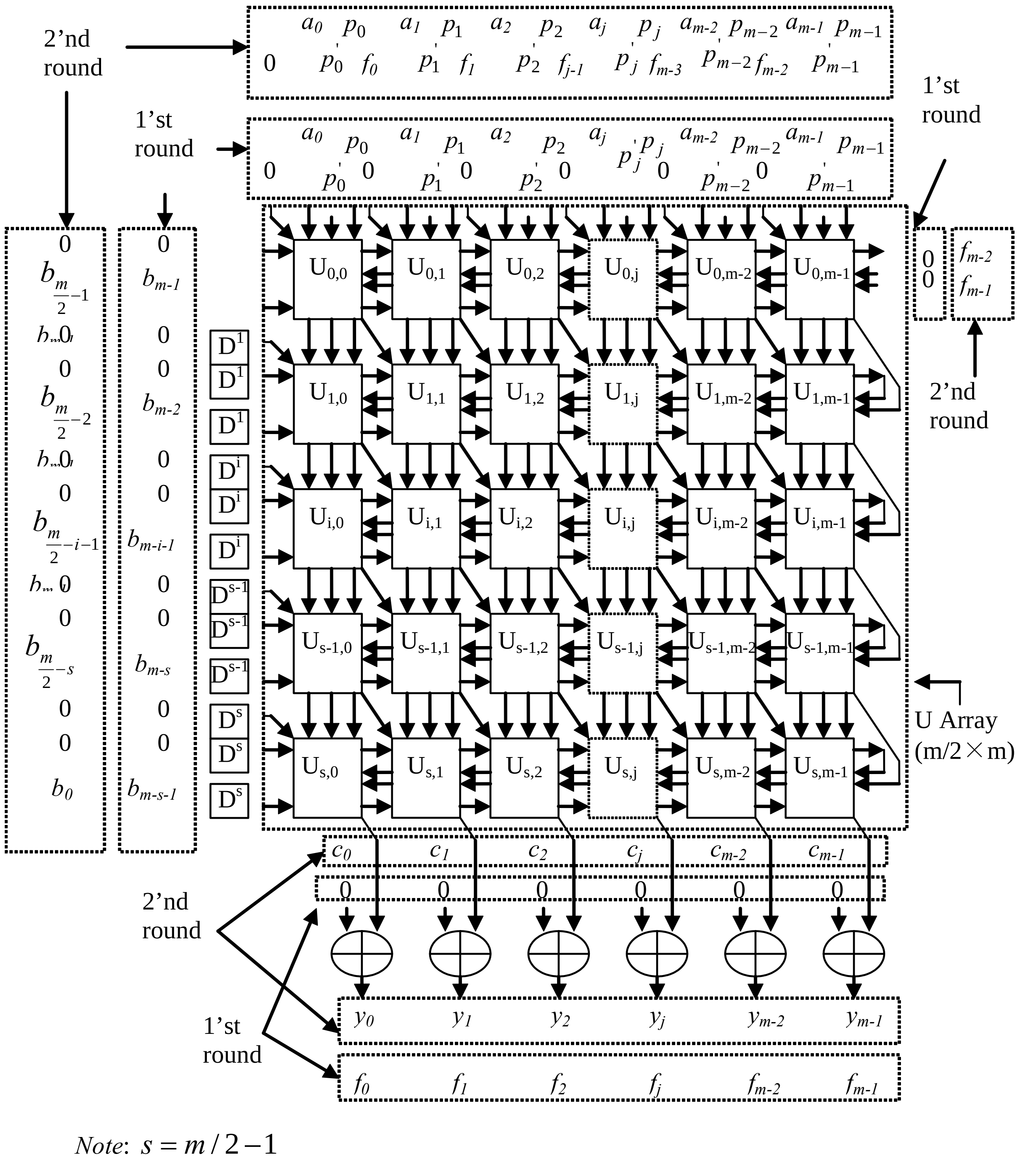

2'nd round
1'st round
U Array (m/2×m)
$U_{0,0}$ $U_{0,1}$ $U_{0,2}$ $U_{0,j}$ $U_{0,m-2}$ $U_{0,m-1}$
$U_{s,0}$ $U_{s,1}$ $U_{s,2}$ $U_{s,j}$ $U_{s,m-2}$ $U_{s,m-1}$
c_0 c_1 c_2 c_j c_{m-2} c_{m-1}
y_0 y_1 y_2 y_j y_{m-2} y_{m-1}
f_0 f_1 f_2 f_j f_{m-2} f_{m-1}
Note: $s = m/2 - 1$

Figure 1: The Semi-systolic Array for Computing $Y=AB^2 + C$.

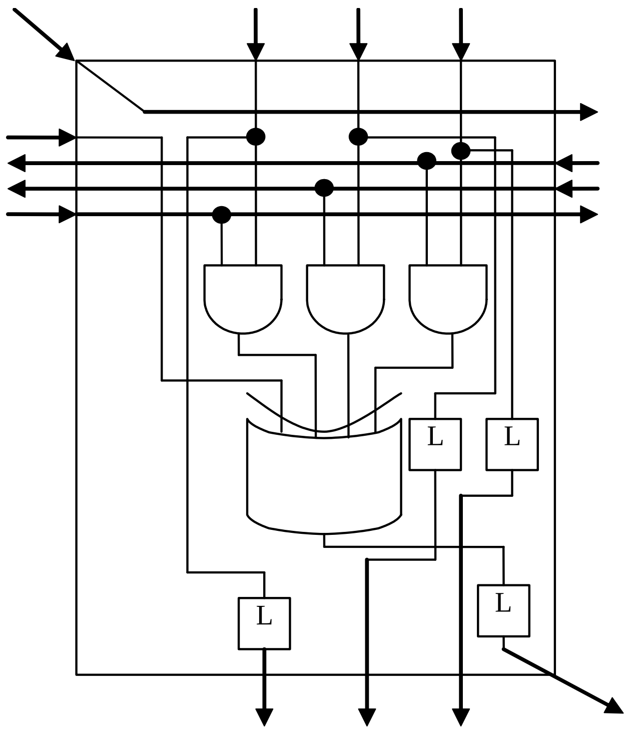


Figure 2: The Detailed Circuit of the Cell U in Fig. 1.

4. THE PROPOSED INVERSION IN GF(2^M)

Most efficient schemes for the inversion in GF(2^m) are mainly based on either Euclid's algorithm or Fermat's theorem. The inversion algorithms based on the Euclid's algorithm usually use the polynomial basis representation and have the benefit of low space complexity [46-49]. The Fermat theorem-based inversion algorithms can use any basis representation, but its best choice is the normal basis representation since the squaring operations in the normal basis representation can be easily implemented by only simply cyclic shifting [50-51]. However, the drawback of the normal basis representation is that it needs basis conversions. For short latency, Wei [17, 52] provides a semi-systolic inversion with the Fermat theorem and the polynomial basis. In this paper, we will present a new low-complexity inversion algorithm based on the Fermat theorem and the circuit folding technique and using the polynomial basis representation for further reducing space cost.

Inversion can be considered as a special case of exponentiation because

$$B^{-1} = B^{2^m-2}$$
$$= B^2 \times (B^2)^2 \times ((B^2)^2)^2 \ldots \times \overbrace{((\ldots((B)^2)^2)^2\ldots)^2}^{m-1}.$$

Table 1
Comparison of Semi-systolic Arrays for Computing AB^2+C in GF(2^m)

Items	*Wei [17]*	*Lee et al. [18]*	*Fig.1*
Function	AB^2+C	AB^2	AB^2+C
Array type	Semi-systolic	Systolic	Semi-systolic
Cell types	1 cell type	3 cell types	1 cell type
No. of cells	m^2	V: $m[m/2]$, W:m, Q:	$m^2/2$
Latency	m cycles	$2m+[m/2]$ cycles	m cycles
Propagation delay per cell	$T_{AND}+T_{XOR3}$	$T_{AND}+T_{XOR2}$	$T_{AND}+T_{XOR4}$
Space complexity	$3m^2$ AND_2	$2m[m/2]$ AND_2	$3m^2/2$ AND_2
	$1m^2$ XOR_2	$2m[m/2]+2m+2[m/2]$ XOR_2	$m^2/2$ XOR_4
	$1m^2$ XOR_3	$7m[m/2]+2m+2[m/2]$ L_1	$2m^2$ L_1
	$4m^2$ L_1	$2m+2[m/2]$ 2×1 Switch	
		m 3×1 Switch	
Transistor count	$68m^2$	$40m^2+71m$	$34m^2$
Algorithm	LSB	MSB	MSB

Note: AND_i: i-input AND gate, XOR_i: i-input XOR gate, L_1: 1-bit latch.
T_{AND}: propagation delay of a 2-input AND gate.
$TXOR_i$: propagation delay of an i-input XOR gate.

The concept of computing multiplicative inverses using consecutive multiplications may be performed under the polynomial basis and the normal basis. Most inverter architectures have been proposed under the normal basis since the squaring operations can be implemented by only simply cyclic shifting. However, existing normal basis multipliers based on the Fermat's theorem use XOR trees for low time complexity are not regular and modular, hence are not suitable for VLSI implementation. Hence, the computation speed of inverters is slower than those of systolic-type inverters under the polynomial basis. For the polynomial basis, it is difficult to compute inverses using Fermat's theorem if the value m of GF(2^m) is large. To overcome this problem, this study presents a low-complexity systolic array inverter by using the circuit folding technique.

The conventional inversion algorithm based on Eq. (2) is described in Algorithm A. Both multiplication operations, Step-A4 and Step-A5, are required in each iteration of the Algorithm A. Suppose that each multiplication operation has the general form $D = H \times K \mod P(\alpha)$. *D, H, K* are any elements in GF(2^m) and are expressed as follows:

$$D = d_0 + d_1\alpha + d_2\alpha^2 + \ldots + d_{m-1}\alpha^{m-1}$$
$$H = h_0 + h_1\alpha + h_2\alpha^2 + \ldots + h_{m-1}\alpha^{m-1}$$
$$K = k_0 + k_1\alpha + k_2\alpha^2 + \ldots + k_{m-1}\alpha^{m-1}$$

where

$$d_i, h_i, k_i \in GF(2) \text{ for } 0 \le \mathrm{i} \le \mathrm{m}\text{-}1.$$

Algorithm A: (Conventional inversion algorithm using Fermat's theorem)

```
/* Computing $Y = B^{-1} \mod P(\alpha)$ */
                Begin
Step-A1: Q:=B;
Step-A2: Y:=1;
Step-A3: For i=1 To m-1 Do
            Begin
Step-A4:        Q:=Q×Q mod P(α);
Step-A5:        Y:=Y×Q mod P(α);
         End;
Step-A6: Return Y;
            End.
```

The multiplication $D = H \times K \mod P(\alpha)$ can be factored as follows:

$$\begin{aligned} D &= H \times K \bmod P(\alpha) \\ &= H \times \left(k_0 + k_1\alpha + k_2\alpha^2 + \ldots + k_{m-1}\alpha^{m-1}\right) \\ &= H \times \begin{pmatrix} \left(k_0 + k_2\alpha^2 + k_4\alpha^4 + \ldots + k_{m-2}\alpha^{m-2}\right) + \\ \alpha\left(k_1 + k_3\alpha^2 + k_5\alpha^4 + \ldots + k_{m-1}\alpha^{m-2}\right) \end{pmatrix} \\ &= H \times \begin{pmatrix} \left(k_0 + k_2\alpha + k_4\alpha^2 + \ldots + k_{m-2}\alpha^{\frac{m}{2}-1}\right)^2 + \\ \alpha\left(k_1 + k_3\alpha + k_5\alpha^2 + \ldots + k_{m-1}\alpha^{\frac{m}{2}-1}\right)^2 \end{pmatrix} \\ &= H \times K_1^2 + \alpha \times \left(H \times K_2^2\right) \end{aligned} \tag{10}$$

where

$$K_1 = k_0 + k_2\alpha + k_4\alpha^2 + \ldots + k_{m-2}\alpha^{\frac{m}{2}-1}$$
$$K_2 = k_1 + k_3\alpha + k_5\alpha^2 + \ldots + k_{m-1}\alpha^{\frac{m}{2}-1}$$

Eq. (10) shows that a multiplication operation can be separated as two power-sum operations. Hence, the proposed power-sum systolic array architecture in the former section can be employed for computing Eq.(10). Both power-sum operations are expressed as follows.

$$\begin{aligned} & H \times K_1^2 \\ & = \left(\left(\left(\left(\left(HK_{1H}\right)\alpha^2 + Hk_{\frac{m}{2}-2}\right)\alpha^2 + Hk_{\frac{m}{2}-4}\right)\alpha^2 + \ldots\right)\alpha^2 + Hk_0\right) \\ & \text{and} \\ & HK_{1H} = \left(\left(\left(\left((0)\alpha^2 + Hk_{m-2}\right)\alpha^2 + Hk_{m-4}\right)\alpha^2 + \ldots\right)\alpha^2 + Hk_{\frac{m}{2}}\right) \end{aligned} \tag{11}$$

where

$$K_{1H} = k_{\frac{m}{2}} + k_{\frac{m}{2}+2}\alpha^2 + k_{\frac{m}{2}+4}\alpha^4 + \ldots + k_{m-2}\alpha^{\frac{m}{2}-2}$$
$$K_{1L} = k_0 + k_2\alpha^2 + k_4\alpha^4 + \ldots + k_{\frac{m}{2}-2}\alpha^{\frac{m}{2}-2}$$

$$\begin{aligned} & H \times K_2^2 \\ & = \left(\left(\left(\left(\left(HK_{2H}\right)\alpha^2 + Hk_{\frac{m}{2}-1}\right)\alpha^2 + Hk_{\frac{m}{2}-3}\right)\alpha^2 + \ldots\right)\alpha^2 + Hk_1\right) \\ & \text{and} \\ & HK_{2H} = \left(\left(\left(\left((0)\alpha^2 + Hk_{m-1}\right)\alpha^2 + Hk_{m-3}\right)\alpha^2 + \ldots\right)\alpha^2 + Hk_{\frac{m}{2}+1}\right) \end{aligned} \tag{12}$$

where

$$K_{2H} = k_{\frac{m}{2}+1} + k_{\frac{m}{2}+3}\alpha^2 + k_{\frac{m}{2}+5}\alpha^4 + \ldots + k_{m-1}\alpha^{\frac{m}{2}-2}$$
$$K_{2L} = k_1 + k_3\alpha^2 + k_5\alpha^4 + \ldots + k_{\frac{m}{2}-1}\alpha^{\frac{m}{2}-2}$$

According to Eqs. (10-12), Fig. 3 shows the systolic array architecture for computing $D = H \times K \bmod P(\alpha)$ by utilizing the circuit folding technique. The U array in Fig. 3 is similar to the kernel U array in Fig. 1. The difference between them is just their sizes. The U array in Fig.1 is an $\frac{m}{2} \times m$ array while that in Fig. 3 is an $\frac{m}{4} \times m$ array. The function block × α modifier is realized by Fig. 4. The 1st round and the 3rd round are responsible for computing $H \times K_1^2 \bmod P(\alpha)$, and the 2nd round and the 4th round are for $H \times K_2^2 \bmod P(\alpha)$, respectively. At the final step, the temporal result of $H \times K_2^{\,2} \bmod P(\alpha)$ is modified by multiplying ± through the function and then is summed to the intermittent result of for the final result D. In Algorithm A, both steps Step-A4 and Step–A5 are time-dependent. In other words, the result of the Step-A4 is the income of the Step-A5. Thus, Algorithm A requires 2m multiplication operations. For further reduction of multiplication operations, the parallel processing concept is employed and the parallel processing version of Algorithm A is depicted in Algorithm B. Both Step-B5 and Step-B6 are performed concurrently.

Algorithm B: (The proposed parallel inversion algorithm using Fermat's theorem)

```
/* Computing $Y = B^{-1} \bmod P(\alpha)$ */
          Begin
Step-B1:Q:=B×B mod P(α);
Step-B2:W=Q;
Step-B3:Y:=1;
Step-B4:For i=1 To m-1 Do
          Begin
          Cobegin
Step-B5:Q:=Q×Q mod P(α);
Step-B6:Y:=Y×W mod P(α);
          Coend
Step-B7:W:=Q;
          End
Step-B8:Return Y;
          End
```

By applying Eqs. (10-12), the multiplication operation Q: = Q × Q mod P(±) in Step-B5 can be factored as follows.

$$Q = Q \times Q \bmod P(\alpha) \tag{13}$$
$$= \left(Q \times Q_1^2 \bmod P(\alpha)\right) + \left(\alpha \times \left(Q \times Q_2^2\right) \bmod P(\alpha)\right)$$

where

$$Q_1 = q_0 + q_2\alpha + q_4\alpha^2 + \ldots + q_{m-2}\alpha^{\frac{m}{2}-1}$$
$$Q_2 = q_1 + q_3\alpha + q_5\alpha^5 + \ldots + q_{m-1}\alpha^{\frac{m}{2}-1}$$

$$Q \times Q_1^2$$
$$= \left(\left(\left(\left(\left(QQ_{1H}\right)\alpha^2 + Qq_{\frac{m}{2}-2}\right)\alpha^2 + Qq_{\frac{m}{2}-4}\right)\alpha^2 + \ldots\right)\alpha^2 + Qq_0\right)$$

and

$$QQ_{1H} = \left(\left(\left(\left((0)\alpha^2 + Qq_{m-2}\right)\alpha^2 + Qq_{m-4}\right)\alpha^2 + \ldots\right)\alpha^2 + Qq_{\frac{m}{2}}\right) \tag{14}$$

where

$$Q_{1H} = q_{\frac{m}{2}} + q_{\frac{m}{2}+2}\alpha^2 + q_{\frac{m}{2}+4}\alpha^4 + \ldots + q_{m-2}\alpha^{\frac{m}{2}-2},$$
$$Q_{1L} = q_0 + q_2\alpha^2 + q_4\alpha^4 + \ldots + q_{\frac{m}{2}-2}\alpha^{\frac{m}{2}-2}.$$

$$Q \times Q_2^2$$
$$= \left(\left(\left(\left(\left(QQ_{2H}\right)\alpha^2 + Qq_{\frac{m}{2}-1}\right)\alpha^2 + Qq_{\frac{m}{2}-3}\right)\alpha^2 + \ldots\right)\alpha^2 + Qq_1\right)$$

and

$$QQ_{2H} = \left(\left(\left(\left((0)\alpha^2 + Qq_{m-1}\right)\alpha^2 + Qq_{m-3}\right)\alpha^2 + \ldots\right)\alpha^2 + Qq_{\frac{m}{2}+1}\right) \tag{15}$$

where

$$Q_{2H} = q_{\frac{m}{2}+1} + q_{\frac{m}{2}+3}\alpha^2 + q_{\frac{m}{2}+5}\alpha^4 + \ldots + q_{m-1}\alpha^{\frac{m}{2}-2},$$
$$Q_{2L} = q_1 + q_3\alpha^2 + q_5\alpha^4 + \ldots + q_{\frac{m}{2}-1}\alpha^{\frac{m}{2}-2}.$$

Similarly, the multiplication operation Y: = Y×W mod P(α) in Step-B6 can be expressed as follows.

$$Y = Y \times W \bmod P(\alpha)$$
$$= \left(Y \times W_1^2 \bmod P(\alpha)\right) + \left(\alpha \times \left(Y \times W_2^2\right) \bmod P(\alpha)\right) \tag{16}$$

where

$$W_1 = w_0 + w_2\alpha + w_4\alpha^2 + \ldots + w_{m-2}\alpha^{\frac{m}{2}-1}$$
$$W_2 = w_1 + w_3\alpha + w_5\alpha^2 + \ldots + w_{m-1}\alpha^{\frac{m}{2}-1}$$

$$Y \times W_1^2$$
$$= \left(\left(\left(\left(\left(YW_{1H}\right)\alpha^2 + Yw_{\frac{m}{2}-2}\right)\alpha^2 + Yw_{\frac{m}{2}-4}\right)\alpha^2 + \ldots\right)\alpha^2 + Yw_0\right)$$

and

$$YW_{1H} = \left(\left(\left(\left((0)\alpha^2 + Yw_{m-2}\right)\alpha^2 + Yw_{m-4}\right)\alpha^2 + \ldots\right)\alpha^2 + Yw_{\frac{m}{2}}\right) \tag{17}$$

where

$$W_{1H} = w_{\frac{m}{2}} + w_{\frac{m}{2}+2}\alpha^2 + w_{\frac{m}{2}+4}\alpha^4 + \ldots + w_{m-2}\alpha^{\frac{m}{2}-2}$$
$$W_{1L} = w_0 + w_2\alpha^2 + w_4\alpha^4 + \ldots + w_{\frac{m}{2}-2}\alpha^{\frac{m}{2}-2}$$

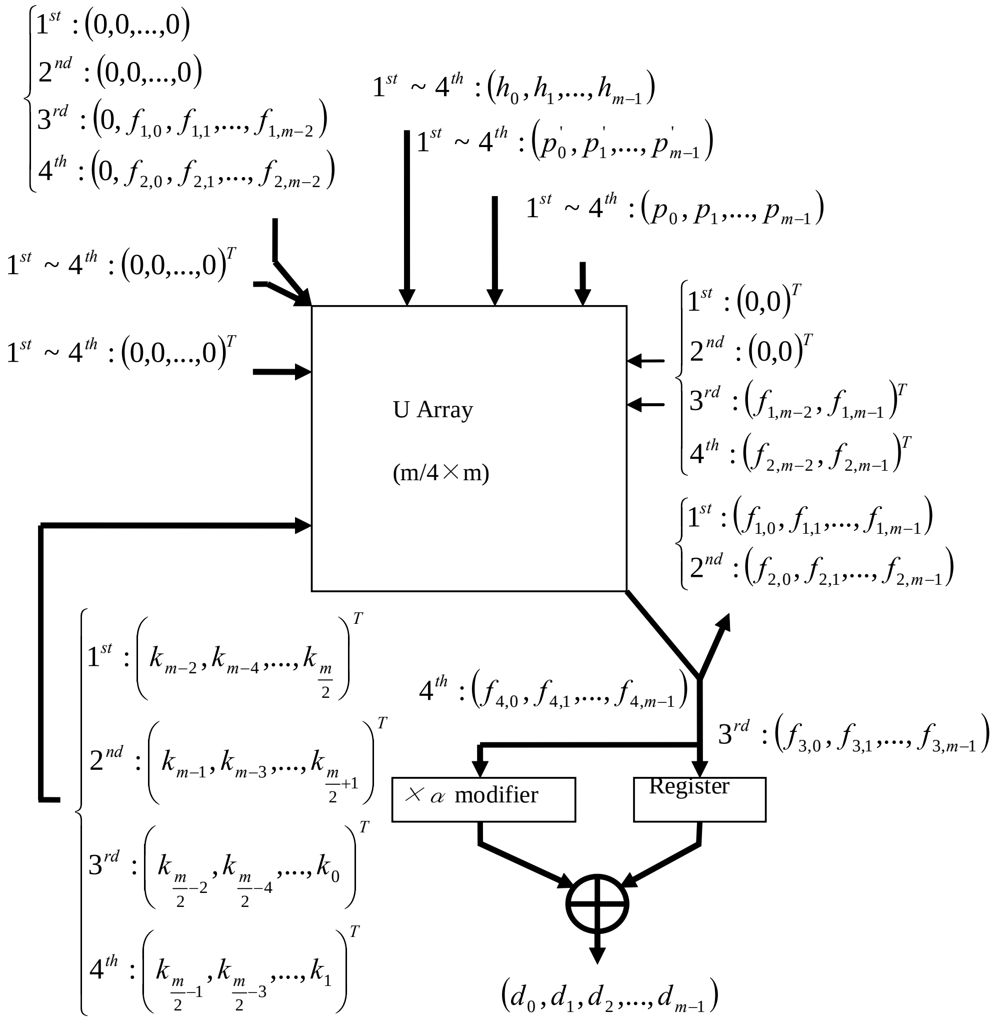

$1^{st}:(0,0,\ldots,0)$
$2^{nd}:(0,0,\ldots,0)$
$3^{rd}:(0,f_{1,0},f_{1,1},\ldots,f_{1,m-2})$
$4^{th}:(0,f_{2,0},f_{2,1},\ldots,f_{2,m-2})$
$1^{st}\sim 4^{th}:(h_0,h_1,\ldots,h_{m-1})$
$1^{st}\sim 4^{th}:(p'_0,p'_1,\ldots,p'_{m-1})$
$1^{st}\sim 4^{th}:(p_0,p_1,\ldots,p_{m-1})$
$1^{st}\sim 4^{th}:(0,0,\ldots,0)^T$
$1^{st}\sim 4^{th}:(0,0,\ldots,0)^T$
U Array
(m/4×m)
$1^{st}:(0,0)^T$
$2^{nd}:(0,0)^T$
$3^{rd}:(f_{1,m-2},f_{1,m-1})^T$
$4^{th}:(f_{2,m-2},f_{2,m-1})^T$
$1^{st}:(f_{1,0},f_{1,1},\ldots,f_{1,m-1})$
$2^{nd}:(f_{2,0},f_{2,1},\ldots,f_{2,m-1})$
$1^{st}:\left(k_{m-2},k_{m-4},\ldots,k_{\frac{m}{2}}\right)^T$
$2^{nd}:\left(k_{m-1},k_{m-3},\ldots,k_{\frac{m}{2}+1}\right)^T$
$3^{rd}:\left(k_{\frac{m}{2}-2},k_{\frac{m}{2}-4},\ldots,k_0\right)^T$
$4^{th}:\left(k_{\frac{m}{2}-1},k_{\frac{m}{2}-3},\ldots,k_1\right)^T$
$4^{th}:(f_{4,0},f_{4,1},\ldots,f_{4,m-1})$
$3^{rd}:(f_{3,0},f_{3,1},\ldots,f_{3,m-1})$
$\times\alpha$ modifier
Register
$(d_0,d_1,d_2,\ldots,d_{m-1})$

Fig. 3. The proposed circuit for realizing *D=H×K mod P(α)*.

Figure 3: The Proposed Circuit for Realizing *D=H×K mod P(α).*

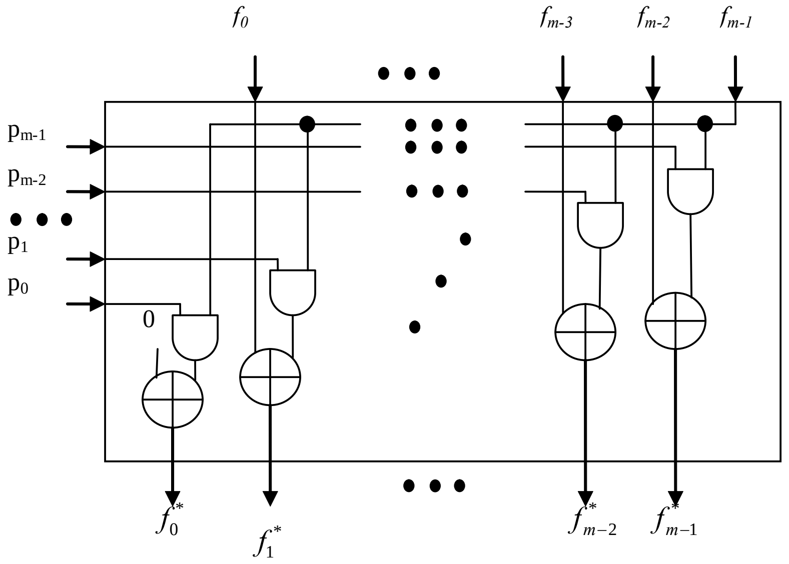


Figure 4: The Detailed Circuit of the Function × α modifier in Fig. 3.

$$
\begin{aligned}
&Y \times W_2^{\,2} \\
&= \left(\left(\left(\left(\left(YW_{2H}\right)\alpha^2 + Yw_{\frac{m}{2}-1}\right)\alpha^2 + Yw_{\frac{m}{2}-3}\right)\alpha^2 + \ldots\right)\alpha^2 + Yw_1\right) \\
&\text{and} \\
&YW_{2H} = \left(\left(\left(\left((0)\alpha^2 + Yw_{m-1}\right)\alpha^2 + Yw_{m-3}\right)\alpha^2 + \ldots\right)\alpha^2 + Yw_{\frac{m}{2}+1}\right)
\end{aligned}
\tag{18}
$$

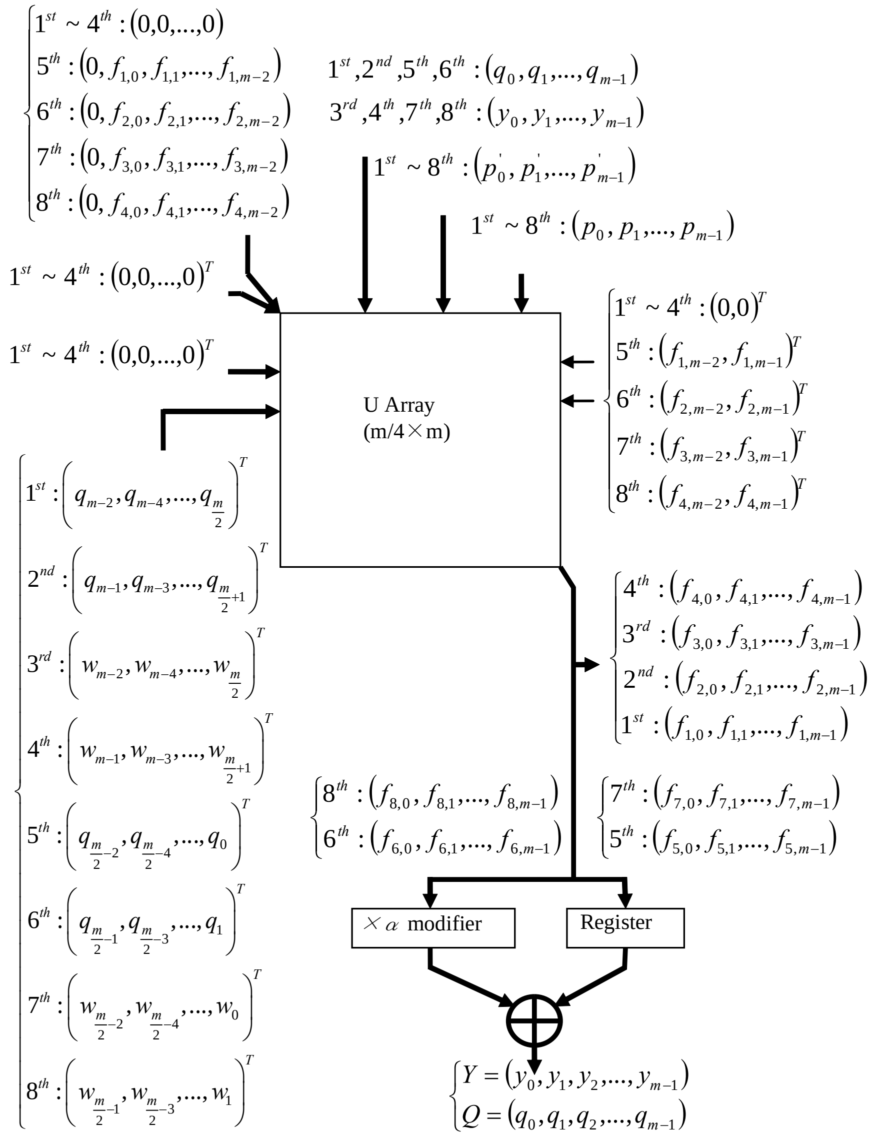
where

$$W_{2H} = w_{\frac{m}{2}+1} + w_{\frac{m}{2}+3}\alpha^2 + w_{\frac{m}{2}+5}\alpha^4 + \ldots + w_{m-1}\alpha^{\frac{m}{2}-2}$$

$$W_{2L} = w_1 + w_3\alpha^2 + w_5\alpha^4 + \ldots + w_{\frac{m}{2}-1}\alpha^{\frac{m}{2}-2}$$

The Algorithm B roughly takes m multiplication execution time. Based on the systolic architecture in Fig. 3, Eqs. (13-18), and the pipelined method, both multiplication operations (Step-B5 and Step-B6) can be performed in parallel and the procedure is shown in Fig. 5. Four rounds, the 1st, the 2nd, the 5th, and the 6th round, are charged with computing Q: = $Q \times Q$ mod $P(\alpha)$ in Step-B5. Another four rounds, the 3rd, the 4th, the 7th, and the 8th round, are responsible for performing. By utilizing the multiplication array in Fig. 5, the execution flow of Algorithm B is described in Fig. 6.

Comparison of various systolic inverters is listed in Table 2. The results show that our proposed systolic inverter using the circuit folding technique saves about 75% space complexity while comparing with existing systolic inverter which is based on Fermat's algorithm. Furthermore, the latency of our proposed inverter takes only $m^2/2$ clock cycles



$1^{st} \sim 4^{th} : (0,0,\ldots,0)$
$5^{th} : (0, f_{1,0}, f_{1,1}, \ldots, f_{1,m-2})$
$6^{th} : (0, f_{2,0}, f_{2,1}, \ldots, f_{2,m-2})$
$7^{th} : (0, f_{3,0}, f_{3,1}, \ldots, f_{3,m-2})$
$8^{th} : (0, f_{4,0}, f_{4,1}, \ldots, f_{4,m-2})$
$1^{st}, 2^{nd}, 5^{th}, 6^{th} : (q_0, q_1, \ldots, q_{m-1})$
$3^{rd}, 4^{th}, 7^{th}, 8^{th} : (y_0, y_1, \ldots, y_{m-1})$
$1^{st} \sim 8^{th} : (p'_0, p'_1, \ldots, p'_{m-1})$
$1^{st} \sim 8^{th} : (p_0, p_1, \ldots, p_{m-1})$
$1^{st} \sim 4^{th} : (0,0,\ldots,0)^T$
$1^{st} \sim 4^{th} : (0,0,\ldots,0)^T$
U Array (m/4×m)
$1^{st} \sim 4^{th} : (0,0)^T$
$5^{th} : (f_{1,m-2}, f_{1,m-1})^T$
$6^{th} : (f_{2,m-2}, f_{2,m-1})^T$
$7^{th} : (f_{3,m-2}, f_{3,m-1})^T$
$8^{th} : (f_{4,m-2}, f_{4,m-1})^T$
$1^{st} : (q_{m-2}, q_{m-4}, \ldots, q_{\frac{m}{2}})^T$
$2^{nd} : (q_{m-1}, q_{m-3}, \ldots, q_{\frac{m}{2}+1})^T$
$3^{rd} : (w_{m-2}, w_{m-4}, \ldots, w_{\frac{m}{2}})^T$
$4^{th} : (w_{m-1}, w_{m-3}, \ldots, w_{\frac{m}{2}+1})^T$
$5^{th} : (q_{\frac{m}{2}-2}, q_{\frac{m}{2}-4}, \ldots, q_0)^T$
$6^{th} : (q_{\frac{m}{2}-1}, q_{\frac{m}{2}-3}, \ldots, q_1)^T$
$7^{th} : (w_{\frac{m}{2}-2}, w_{\frac{m}{2}-4}, \ldots, w_0)^T$
$8^{th} : (w_{\frac{m}{2}-1}, w_{\frac{m}{2}-3}, \ldots, w_1)^T$
$4^{th} : (f_{4,0}, f_{4,1}, \ldots, f_{4,m-1})$
$3^{rd} : (f_{3,0}, f_{3,1}, \ldots, f_{3,m-1})$
$2^{nd} : (f_{2,0}, f_{2,1}, \ldots, f_{2,m-1})$
$1^{st} : (f_{1,0}, f_{1,1}, \ldots, f_{1,m-1})$
$8^{th} : (f_{8,0}, f_{8,1}, \ldots, f_{8,m-1})$
$6^{th} : (f_{6,0}, f_{6,1}, \ldots, f_{6,m-1})$
$7^{th} : (f_{7,0}, f_{7,1}, \ldots, f_{7,m-1})$
$5^{th} : (f_{5,0}, f_{5,1}, \ldots, f_{5,m-1})$
$\times\alpha$ modifier
Register
$Y = (y_0, y_1, y_2, \ldots, y_{m-1})$
$Q = (q_0, q_1, q_2, \ldots, q_{m-1})$

Figure 5: The Proposed Circuit for Concurrently Realizing both Q: = $Q \times Q$ mod $P(\alpha)$ and Y: = $Y \times W$ mod $P(\alpha)$.

while the traditional inverters may need at least m(m-1) clock cycles. In other words, our proposed inverter takes only 50% execution time of that of other existing one based on Fermat's algorithm. Our proposed systolic inverter is slower than the Yan-Sarwate-Liu inverter [49] which is based on the modified Euclidean algorithm. However, the proposed inverter saves about 93% space complexity as compared to the Yan-Sarwate-Liu inverter. The proposed inverter is suitable for the resource constrained devices such as portable devices (i.e, PDAs, smart phones).

In summary, our proposed inverter using the circuit folding technique saves both space and time complexities as compared to other existing traditional inverters based on the same Fermat algorithm.

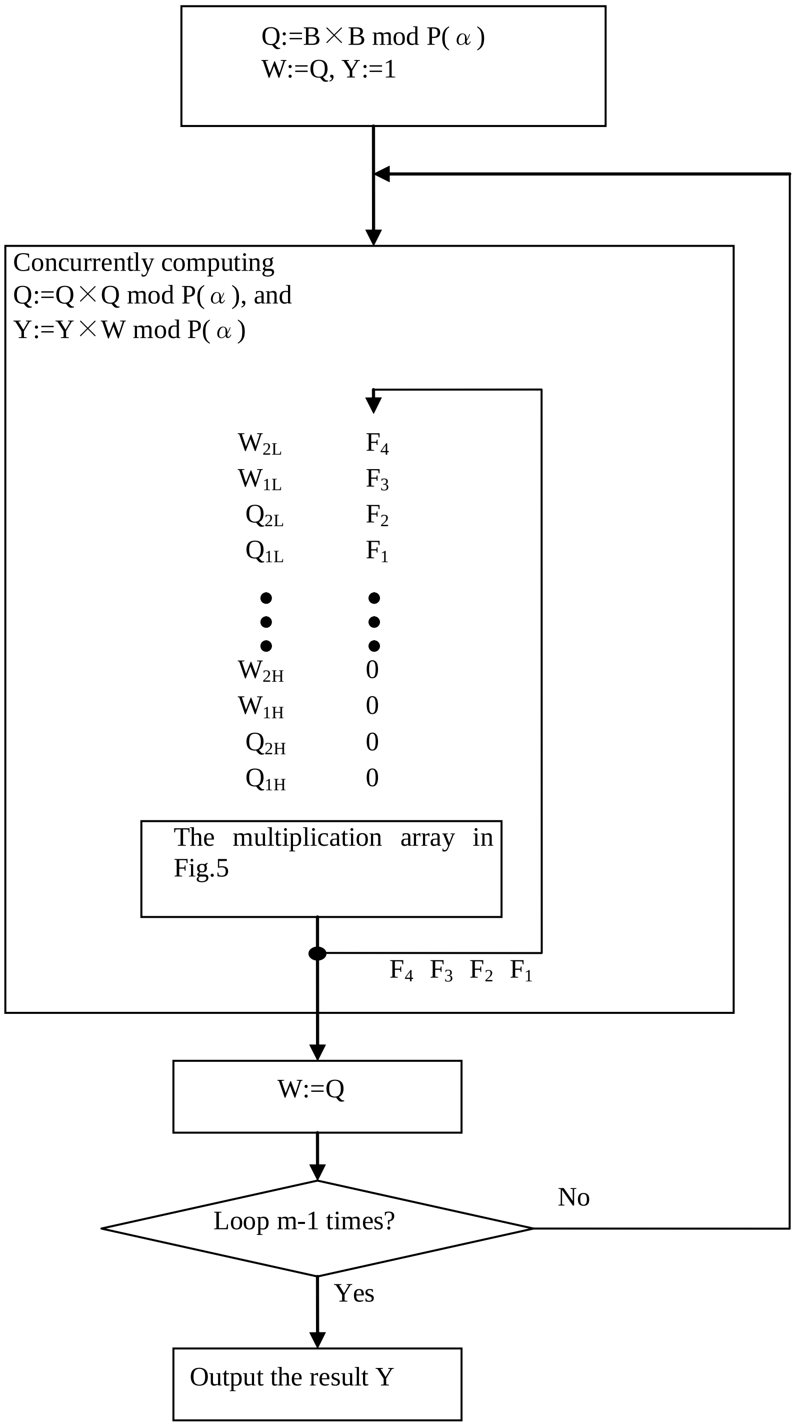


Figure 6: The Execution Flow for Computing $Y=B^{-1} \bmod P(\alpha)$ by Using the Multiplication Array in Fig. 5.

Table 2
Comparison of Semi-systolic Arrays for Computing Inversion in GF(2^m)

Items	*Yan et al. [49]*	*Wei [17]*	*Proposed in Fig.5*
Basis	Polynomial	Polynomial	Polynomial
Algorithm	Euclid	Fermat	Fermat
No. of cells	Cell type-4: (2m-1)×m Cell type-5: 2m-1	m^2	$m^2/4$
No. of cell types	2	1	1
Latency (unit=cycles)	5m-2	m(m-1)	$m^2/2$
Propagation delay per cell	$T_{AND}+T_{MUX}$	$T_{AND}+T_{XOR3}$	$T_{AND}+T_{XOR4}$
Cell complexity	3m(2m-1) AND_2 2m(2m-1) XOR_2 ($6m^2$-m-1) 2-to-1 MUX$22m^2$-9m-2 L_1	$3m^2$ AND_2 $1m^2$ XOR_2 $1m^2$ XOR_3 $4m^2$ L_1	$3m^2/4$ AND_2 $m^2/4$ XOR_4 m^2 L_1
Transistor count	$244m^2$	$68m^2$	$17m^2$
Algorithm	MSB	MSB	MSB

Note: AND_i: i-input AND gate, XOR_i: i-input XOR gate, L_1: 1-bit latch

5. THE PROPOSED DIVISION IN GF(2^M)

Observing Eq.(3), the division operation is similar to the inversion. The division operation A / B is actually equivalent to the multiplication of A and B^{-1}. Thus, the difference between the division and inversion operations is that the initial values for both operations are different. Therefore, the division algorithms based on Algorithm A and Algorithm B are rewritten as follows:

Algorithm C: (Conventional division algorithm using Fermat's theorem)

```
/* Computing $Y = \frac{A}{B} \bmod P(\alpha)$ */
    Begin
Step-C1:  Q:=B;
Step-C2:  Y:=A;
Step-C3:  For i=1 To m-1 Do
    Begin
Step-C4:  Q:=Q×Q mod P(α);
Step-C5:  Y:=Y×Q mod P(α);
    End
Step-C6:  Return Y;
    End.
```

Algorithm D: (Parallel division algorithm using Fermat's theorem)

```
/* Computing $Y = \frac{A}{B} \bmod P(\alpha)$ */
    Begin
Step-D1:  Q:=B×B mod P(α);
Step-D2:  W=Q;
Step-D3:  Y:=A;
Step-D4:  For i=1 To m-1 Do
    Begin
        Cobegin
Step-D5:  Q:=Q×Q mod P(α);
Step-D6:  Y:=Y×W mod P(α);
          Coend
Step-D7:  W:=Q;
          End
Step-BD8: Return Y;
    End
```

Algorithm C shows the traditional division algorithm and Algorithm D is the proposed parallel division algorithm. The proposed inversion architecture in Fig.5 is also useful for the proposed division architecture. Therefore, the proposed division architecture saves 75% space complexity and 50% time complexity while comparing with other existing systolic division architectures.

6. CONCLUSIONS

A new systolic power-sum circuit using the circuit folding technique has been presented herein. The proposed power-sum circuit saves about 50% space complexity and same time complexity while comparing with other existing semi-systolic power-sum circuits. Based on the proposed power-sum circuit, a new efficient systolic inversion architecture has also proposed. As compared to traditional inversion circuits [17] which are based on the same Fermat algorithm, the proposed inversion circuit saves about 75% space complexity and 50% time complexity. Furthermore, the proposed systolic division architecture also saves about 75% space complexity and 50% time complexity while comparing with other existing conventional division circuits [17]. By employing the circuit folding technique, parallel processing, and pipeline processing, our proposed power-sum, inversion, and division circuits provide efficient array architectures for saving both space and time complexities.

ACKNOWLEDGMENTS

The work was supported in part by the National Science Council of Republic of China under grant number NSC 95-2221-E-231-019.

REFERENCES

[1] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam: North-Holland, 1977.

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

[3] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*, Reading, Mass.: Addison-Wesley, 1985.

[4] I. S. Reed and T. K. Truong, "The use of Finite Fields to Compute Convolutions", *IEEE Trans. Information Theory*, Vol. IT-21, No. 2, March 1975, pp. 208-213.

[5] B. Benjauthrit and I. S. Reed, "Galois Switching Functions and their Applications", *IEEE Trans. Computers*, Vol. C-25, Jan. 1976, pp. 78-86.

[6] C. C. Wang and D. Pei, "A VLSI Design for Computing Exponentiation in GF(2^m) and its Application to Generate Pseudorandom Number Sequences", *IEEE Trans. Computers*, Vol. 39, No. 2, Feb. 1990, pp. 258-262.

[7] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions in Information Theory*, Vol.I T-22, No. 6, Nov. 1976, pp. 644-654.

[8] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", *Communications of the ACM*, Vol. 21, No. 2, Feb. 1978, pp. 120-126.

[9] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press: New York, 1999.

[10] H. Okano and H. Imai, "A Construction Method of High-Speed Decoders using ROM's for Bose-Chaudhuri-Hocquenghem and Reed-Solomon Codes", *IEEE Trans. Computers*, Vol. C-36, 1987, pp. 1165-1171.

[11] C. L. Wang and W. J. Bair, "A VLSI Architecture for Implementation of the Decoder for Binary BCH Codes", in *Proc. Symp. Commun.*, Taiwan, Dec. 1991, pp. 36-40.

[12] S.W. Wei and C.H. Wei, "High Speed Decoder of Reed-Solomon Codes", *IEEE Trans. Commun.*, Vol. 41, No. 11, Nov. 1993, pp. 1588-1593.

[13] S.W. Wei, "VLSI Architectures for Computing Exponentiations, Multiplicative Inverses, and Divisions in GF(2^m)", *IEEE Trans. Circuits Syst.*, Vol. 44, 1997, pp. 847-855.

[14] S.W. Wei, "A Systolic Power-sum Circuit for GF(2^m)", *IEEE Trans. Computers*, Vol. 43, No. 2, Feb. 1994, pp. 226-229.

[15] C.L. Wang and J.H. Guo, "New Systolic Arrays for C+AB^2, Inversion, and Division in GF(2^m)", *IEEE Trans. Computers*, Vol. 49, No. 10, Oct. 2000, pp. 1120-1125.

[16] N.Y. Kim, H.S. Kim, and K.Y. Yoo, "Computation of AB^2 Multiplication in GF(2^m) using Low-complexity Systolic Architecture", *IEE Proc.-Circuits Devices Syst.*, Vol. 150, No. 2, April 2003, pp. 119-123.

[17] S.W. Wei, "VLSI Architectures for Computing Exponentiations, Multiplicatives Inverses, and Divisions in GF(2^m)", *IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 44, No. 10, Oct. 1997, pp. 847-855.

[18] C.Y. Lee, C.W. Chiou, A.W. Deng, and J.M. Lin, "Low-complexity Bit-parallel Systolic Architectures for Computing A(x)B^2(x) over GF(2m)", *IEE Proc.-Circuits Devices Syst.*, Vol. 153, No. 4, August 2006, pp. 399-406.

[19] "The Digital Signature Standard Proposed by NIST", *Communication of ACM*, July 1992, Vol. 35, No. 7, pp. 36-40.

[20] J.H. Guo and C.L. Wang, "Systolic Array Implementation of Euclid's Algorithm for Inversion and Division in GF(2^m)", *IEEE Trans. Computers*, Vol. 47, No. 10, 1998, pp. 1161-1167.

[21] A. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptology*, CRC Press, New York, 1997.

[22] R. Kumanduri and C. Romero, *Number Theory with Computer Applications*, Upper Saddle River, NT: Prentice Hall, 1998.

[23] T. C. Bartee and D. J. Schneider, "Computation with Finite Fields", *Information and Computing*, Vol. 6, Mar. 1963, pp. 79-98.

[24] E. D. Mastrovito, "VLSI Architectures for Multiplication over Finite Field GF(2^m)", *Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, Proc. Sixth Int'l Conf.*, AAECC-6, T. Mora, *ed.*, Rome, July 1988, pp. 297-309.

[25] E. D. Mastrovito, "VLSI Architectures for Computations in Galois Fields", Ph.D. thesis, Linköping Univ., Dept. of Electrical Eng., Linköping, Sweden, 1991.

[26] Ç. K. Koç and B. Sunar, "Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields", *IEEE Trans. Computers*, Vol. 47, No. 3, March 1998, pp. 353-356.

[27] C. Y. Lee, "Low Complexity Bit-parallel Systolic Multiplier over GF(2^m) Using Irreducible Trinomials", *IEE Proc.-Comput. Digit. Tech.*, Vol. 150, No. 1, Jan. 2003, pp. 39-42.

[28] T. Itoh and S. Tsujii, "Structure of Parallel Multipliers for a Class of Fields GF(2^m)", *Information and Computation*, Vol. 83, 1989, pp. 21-40.

[29] M. A. Hasan, M. Wang, V. K. Bhargava, "Modular Construction of Low Complexity Parallel Multipliers for a Class of Finite Fields GF(2^m)", *IEEE Trans. Computers*, Vol. 41, No. 8, August 1992, pp. 962-971.

[30] 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*, Vol. 50, No. 5, May 2001, pp. 385-393.

[31] C. Paar, "A New Architecture for a Parallel Finite Field Multiplier with low Complexity based on Composite Fields", *IEEE Trans. Computers*, Vol. 45, No. 7, July 1996, pp. 856-861.

[32] C. Paar, P. Fleischmann, and P. Roelse, "Efficient Multiplier Architectures for Galois Fields GF(2^{4n})", *IEEE Trans. Computers*, Vol. 47, No. 2, Feb. 1998, pp. 162-170.

[33] C.W. Chiou, L.C. Lin, F.H. Chou, and S.F. Shu, "Low Complexity Finite Field Multiplier using Irreducible Trinomials", *Electronics Letters*, Vol. 39, No. 24, 27th Nov. 2003, pp. 1709-1711.

[34] J. L. Massey and J. K.Omura, Computational Method and Apparatus for Finite Field Arithmetic, U.S. Patent Number 4, 587, 627, May 1986.

[35] C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)", *IEEE Trans. Computers*, Vol. C-34, No. 8, Aug. 1985, pp. 709-717.

[36] A. Reyhani-Masoleh and M. A. Hasan, "A New Construction of Massey-Omura Parallel Multiplier Over GF(2^m)", *IEEE Trans. Computers*, Vol. 51, No. 5, May 2002, pp. 511-520.

[37] A. Reyhani-Masoleh and M. A. Hasan, "Fast Normal Basis Multiplication using General Purpose Processors", *IEEE Trans. Computers*, Vol. 52, No. 11, Nov. 2003, pp. 1379-1390.

[38] S. Oh, C. H. Kim, J. Lim, and D. H. Cheon, "Efficient Normal Basis Multipliers in Composite Fields", *IEEE Trans. Computers*, Vol. 49, No. 10, Oct. 2000, pp. 1133-1138.

[39] H. Fan and Y. Dai, "Key Function of Normal Basis Multipliers in GF(2^n)", *Electronics Letters*, Vol. 38, No. 23, 7th Nov. 2002, pp. 1431-1432.

[40] B. Sunar and Ç. K. Koç, "An Efficient Optimal Normal Basis type II Multiplier", *IEEE Trans. Computers*, Vol. 50, No. 1, Jan. 2001, pp. 83-87.

[41] N. Takagi, J.-I. Yoshiki, and K. Takagi, "A Fast Algorithm for Multiplicative Inversion in GF(2^m) Using Normal Basis", *IEEE Trans. Computers*, Vol. 50, No. 5, May 2001, pp. 394-398.

[42] E. R. Berlekamp, "Bit-serial Reed-Solomon Encoders", *IEEE Trans. Information Theory*, Vol.I T-28, 1982, pp. 869-874.

[43] H. Wu, M. A. Hasan, and I. F. Blake, "New low-Complexity Bit-parallel Finite Field Multipliers using Weakly Dual Bases", *IEEE Trans. Computers*, Vol. 47, No. 11, November 1998, pp. 1223-1234.

[44] H. Wu and M. A. Hasan, "Low Complexity Bit-parallel Multipliers for a Class of Finite Fields", *IEEE Trans. Computers*, Vol. 47, No. 8, Aug. 1998, pp. 883-887.

[45] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective*, Reading, Mass.: Addison-Wesley, 1985.

[46] H. Brunner, A. Curiger, and M. Hofstetter, "On Computing Multiplicative Inverses in GF(2^m)", *IEEE Trans. Computers*, Vol. 42, No. 8, Aug. 1993, pp. 1010-1015.

[47] J.H. Guo and C.L. Wang, "Hardware-efficient systolic architecture for inversion and division in GF(2^m)", *IEE Proc.-Comput. Digit. Tech.*, Vol. 145, No. 4, July 1998, pp. 272-278.

[48] J.H. Guo and C.L. Wang, "Systolic Array Implementation of Euclid's Algorithm for Inversion and Division in GF(2^m)", *IEEE Trans. Computers*, Vol. 47, No. 10, Oct. 1998, pp. 1161-1167.

[49] Z. Yan, D.V. Sarwate, Z. Liu, "High-speed Systolic Architectures for Finite Field Inversion", Integration, the *VLSI Journal*, January 2005, Vol. 38, No.3, pp. 383-398.

[50] C.C. Wang, T.K. Truong, H.M. Shao, L.J. Deutsch, J.K. Omura, I.S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)", *IEEE Trans. Computers*, Vol. C-34, No. 8, Aug. 1985, pp. 709-717.

[51] N. Takagi, J. Yoshiki, K. Takagi, "A Fast Algorithm for Multiplicative Inversion in GF(2^m) using Normal Basis", *IEEE Trans. Computers*, Vol. 50, No. 5, May 2001, pp. 394-398.

[52] S.W. Wei, "VLSI Architectures for Computing Exponentiations, Multiplicative Inverses, and Divisions in GF(2^m)", in *Proc. 1995 IEEE Int'l Symp. Circuits and Systems*, May 1995, pp. 4.203-4.206.