Bit-Serial Systolic Array Implementation of Euclid's Algorithm for Inversion and Division in $GF(2^m)$

Jyh-Huei Guo and Chin-Liang Wang

Department of Electrical Engineering, National Tsing Hua University
Hsinchu, Taiwan 300, Republic of China

Abstract

This paper presents two serial-in serial-out systolic arrays for inversion or division in $GF(2^m)$ with the standard basis representation. They can produce results at a rate of one per m cycles after an initial delay of $5m - 4$ cycles. The proposed arrays involve unidirectional data flow and are highly regular and modular. Thus, they are well suited to VLSI implementation with fault-tolerant design. As compared to existing related systolic designs with the same time complexity and I/O format, our proposed arrays gain a significant improvement in hardware area.

I. Introduction

Finite fields $GF(2^m)$ have found various applications in areas of communications, such as error-correcting codes [1]-[2] and cryptography [3]. In these applications, computing inverses or divisions in $GF(2^m)$ is usually required. It is thus desirable to design hardware-efficient architectures for them.

A number of VLSI architectures for computing inverses and/or divisions in $GF(2^m)$ have been reported in the literature. Among them, the circuits in [4]-[11] are designed based on the concepts of systolic architecture [12]. As described in [12], a systolic system gains a considerable speed improvement over a von Neumann system and is particularly suitable for VLSI implementation due to its simple, regular communication and control structure. The time complexity is $O(1)$ for the divider in [4] (with $O(m \cdot 2^m)$ area complexity), the dividers in [5]-[6] (with $O(m^3)$ area complexity), the divider in [7] (with $O(m^2)$ area complexity), and the divider and inverter with $O(m^2)$ area complexity in [8]; while the time complexity is $O(m)$ for the dividers in [9]-[11] (with $O(m^2)$ area complexity) and the divider and inverter with $O(m)$ area complexity in [8]. It should be noted that the circuits in [4]-[8] are parallel-in parallel-out schemes, while those in [9]-[11] are serial-in serial-out schemes.

For some applications, where the value of m is large (for example, considering a cryptographic system with $m = 1000$), a system with large area complexity will become very impractical due to its large hardware area. For the situation, a system with small area complexity will be highly desirable. From above, we can see that the divider and inverter with $O(m)$ area complexity in [8] have the least area complexity and area-time product. However, they are both parallel-in parallel-out schemes, where the I/O pin count is too large for large m, and involve bi-directional data flow. They have a throughput rate of producing results at a rate of one per $2m - 2$ cycles, which may be too slow for some applications. Besides, the input data should be interleaved to reach 100% utilization efficiency.

In this paper, two serial-in serial-out systolic arrays are proposed for computing inverses or divisions in $GF(2^m)$. One is for division and the other one is for inversion. Both proposed architectures have $O(m)$ area complexity and $O(m)$ time complexity. If the input data come in continuously, they can produce output results at a rate of one per m cycles after an initial delay of $5m - 4$ cycles. The proposed arrays are highly regular and modular and involve unidirectional data flow. As described in [13]-[14], a system with unidirectional data flow outperforms a system with bi-directional data flow in terms of chip cascadability, fault-tolerant, and possible wafer-scale integration. Thus, they are well suited to VLSI implementation with fault-tolerant design [13]-[14]. As compared to existing related systolic designs with the same time complexity and I/O format, our proposed arrays gain a significant improvement in hardware area.

II. An Algorithm for Inversion / Division in $GF(2^m)$ [8]

Let $A(x)$ and $B(x)$ be two elements in $GF(2^m)$, $G(x)$ be the primitive polynomial used to generate the field, and $C(x)$ be the result of $A(x) / B(x) \bmod G(x)$, where

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_0 \quad (1)$$

$$B(x) = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_0 \quad (2)$$

$$G(x) = x^m + g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_0 \quad (3)$$

$$C(x) = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_0 \quad (4)$$

Each coefficient of the polynomials is in $\{0, 1\}$. To compute the inversion $1 / B(x) \bmod G(x)$, the *Euclid's algorithm* [2] can be used:

Euclid's algorithm for inversion in $GF(2^m)$
$R = B(x)$; $S = G(x)$; $U = 1$; $V = 0$;
while $R \neq 0$, **do**
$Q = S$ DIV R; (* DIV: polynomial division *)
temp $= S + Q \cdot R$; $S = R$; $R =$ temp;
temp $= V + Q \cdot U$; $V = U$; $U =$ temp;
end (* V has the result of $1 / B(x) \bmod G(x)$ *)

One disadvantage of the algorithm is that it does not involve a fixed number of iterations for computing inverses in a given field. This makes it not easily realized using VLSI techniques. In [8], a new variant of Euclid's algorithm for computing divisions in $GF(2^m)$ was proposed. The algorithm consists of $2m - 2$ iterations and can be summarized as follows:

Algorithm for division in $GF(2^m)$ [8]
$R = B(x)$; $S = G = G(x)$; $U = A(x)$; $V = T = 0$;
state = 0; count = 0;
for $i = 1$ to $2m - 2$ **do**

$R = x \cdot R$; $T = x \cdot T \bmod G$; (*)
if state = = 0 **then**
count = count + 1;
if r_m = = 1 **then** (* r_m: coefficient of x^m of R *)
tmp = R; (I)
$R = R + S$; (II)
S = tmp; $T = U$; (I)
state = 1;
end
else
count = count − 1;
if r_m = = 1 **then**
$R = R + S$; $T = T + U$; (II)
end
if count = = 0 **then**
$V = T + V$; $U \leftrightarrow V$; (III)
state = 0;
end
end
end (* U has the result $C(x)$ *)

The algorithm can be used to compute the inversion $1 / B(x)$ mod $G(x)$ simply by setting the initial condition of U to "U = 1", instead of "$U = A(x)$". Besides, the operation "$T = x \cdot T$ mod G" in (*) can be simplified to "$T = x \cdot T$" while computing inversion in $GF(2^m)$.

III. VLSI Architecture and Chip Implementation

A. A Dependence Graph for Inversion/Division in $GF(2^m)$ [8]

Fig. 1 shows a dependence graph (DG) of the above division algorithm in $GF(2^m)$, where $m = 3$ [8]. It consists of $2m - 2$ Type-1 cells and $(2m - 2) \times m$ Type-2 cells, where the functions of these two types of basic cells are depicted in Figs. 2-3. The cells in the ith row of this array realizes the ith iteration. For each row, Type-1 cell generates the control signals Ctrl1, Ctrl2, and Ctrl3 for the present iteration as well as computes the values of count and state for the next iteration (i.e., count′ and state′ in Fig. 2) according to the following logic functions:

$$\text{Ctrl1} = (\text{state} == 0) \& (r_m == 1) \quad (5)$$

$$\text{Ctrl2} = (r_m == 1) \quad (6)$$

$$\text{Ctrl3} = (\text{state} == 1) \& (\text{count} == 0) \quad (7)$$

$$\text{count}' = \begin{cases} \text{count} + 1, & \text{if state} == 0 \\ \text{count} - 1, & \text{if state} == 1 \end{cases} \quad (8)$$

$$\text{state} = \overline{\text{state}}, \text{ if } \begin{cases} ((r_m == 1) \& (\text{state} == 0)) \\ \text{or } ((\text{count} == 0) \& (\text{state} == 1)) \end{cases} \quad (9)$$

Type-2 cells in the corresponding row receive Ctrl1, Ctrl2, and Ctrl3 from Type-1 cell and execute the operations of (I), (II), and (III) when the control signals Ctrl1, Ctrl2, and Ctrl3 are true, respectively, where the $(i + 1)$th Type-2 cell $(0 \le i \le m - 1)$ from the right evaluates the $(i + 1)$th least significant coefficients of R, S, U, V, and T (i.e., r'_i, s'_i, u'_i, v'_i, and t'_i in Fig. 3). The coefficients of division result $C(x)$ will emerge from the bottom row of the array (i.e., after $2m - 2$ iterations).

B. A New Variant of the DG in Section III.A

By projecting the DG in Fig. 1 along the east direction, a one-dimensional signal flow graph (SFG) array can be derived. One disadvantage associated with the SFG array is that, owing to the fact that each Type-1 cell in Fig. 1 involves one adder (subtractor) circuit, each basic cell of the SFG array will involve one adder (subtractor) circuit of $\log_2(m + 1)$ bits and thus has an area complexity of $O(\log_2 m)$. When the value of m gets large, each basic cell will become large. To prevent such a problem, we can adopt the following approach.

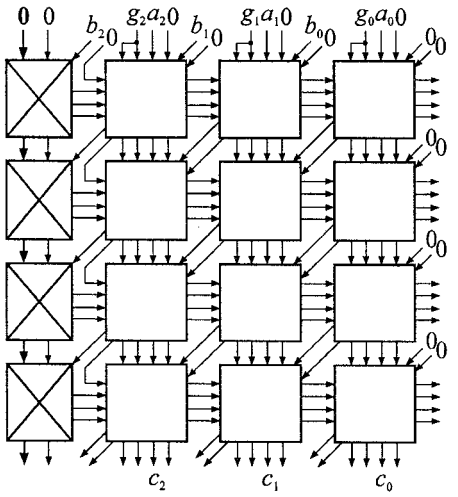

Fig. 1. A dependence graph for division in $GF(2^3)$ [8].

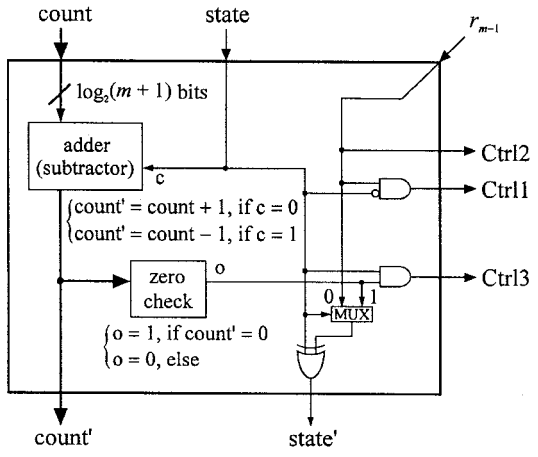

Fig. 2. The circuit of Type-1 cell in Fig .1.

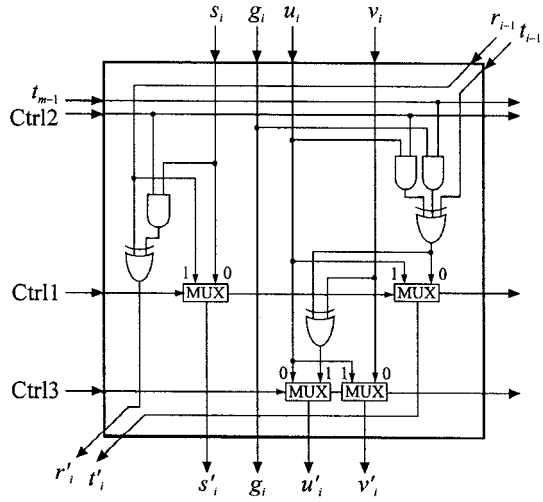

Fig. 3. The circuit of Type-2 cell in Fig. 1.

Fig. 4 shows a new variant of the DG in Fig. 1, where $2m - 2$ Type-3 cells, illustrated in Fig. 5, and $(2m - 2) \times m$ Type-4 cells, depicted in Fig. 6, are used. The functions of Type-3 and Type-4 cells in Fig. 4 are similar to those of Type-1 and Type-2 cells in Fig. 1. The main difference is that each Type-3 cell, not like the Type-1 cell, doesn't involve the adder (subtractor) circuit of $\log_2(m + 1)$ bits. For each row of the DG, Type-3 cell generates the control signals Ctrl2 and Ctrl3 for the present iteration as well as computes the value of state for next iteration (i.e., state′ in Fig. 5). Type-4 cells in the corresponding row receive Ctrl2, Ctrl3, and state from Type-3 cell and generate the control signal Ctrl1 according to the logic function

$$\text{Ctrl1} = (\text{state} == 0) \& \text{Ctrl2} \quad (10)$$

The control signals Ctrl1, Ctrl2, and Ctrl3 are used to control the Type-4 cells in the same way as they are used to control the Type-2 cells described in Section III.A. The method for tracing the value of count can be explained as follows. Each Type-4 cell incorporates extra one 2-to-1 multiplexer for tracing the value of count. For the Type-4 cells in the first row of the DG, only the (1, 1) cell receives 1 for the inc, while the others receive 0's for their inc's. All the cells in the first row receive 0's for their dec's.

1) For the first row of the DG: Since state = 0, the count-flag (c-flag) will be 1 for the (1, 1) cell and 0 for the other Type-4 cells. The count-zero (c-zero) is 0 for the Type-3 cell. Let this indicate that the count is 1 after 1 iteration.
2) For the second row of the DG: If state = 0, the c-flag will be 1 for the (2, 2) cell and 0 for the other Type-4 cells and the c-zero will be 0 for the Type-3 cell. Let this mean that the count becomes 2 after 2 iterations. If state = 1, the c-flag will be 0 for all the Type-4 cells and the c-zero will be 1 for the Type-3 cell. Let this represent that the value of count reduces to 0 after 2 iterations.

It can be checked that, for a certain row of the DG, one of the following two situations occurs.

3) The c-flag is 1 for only one Type-4 cell and 0 for the other Type-4 cells and the c-zero is 0 for the Type-3 cell.
4) The c-zero is 1 for the Type-3 cell and the c-flag is 0 for all the Type-4 cells.

In general, the situation that the c-flag is 1 for the (i, j) cell indicates that the value of count becomes j after i iterations, while the situation that the c-zero is 1 for the Type-3 cell in the ith row means that the value of count reduces to 0 after i iterations. If the c-flag is 1 for the (i, j) cell, where $2 \le j \le m$, the c-flag will be 1 for the $(i + 1, j + 1)$ cell (i.e., count increasing) or $(i + 1, j - 1)$ cell (i.e., count decreasing) depending on what the value of state is. If the c-flag is 1 for the $(i, 1)$ cell, the c-flag will be 1 for the $(i + 1, 2)$ cell for state = 0 or the c-zero will be 1 for the Type-3 cell in the $(i + 1)$th row for state = 1. Besides, if the c-zero is 1 for the Type-3 cell in the ith row, the c-flag will be 1 for the $(i + 1, 1)$ cell.

C. Bit-Serial Systolic Array Implementation

By projecting the DG in Fig. 4 along the east direction following the projection procedure in [15], we can derive a one-dimensional signal flow graph (SFG) array as shown in Fig. 7, where $2m - 2$ basic cells as shown in Fig. 8 are used

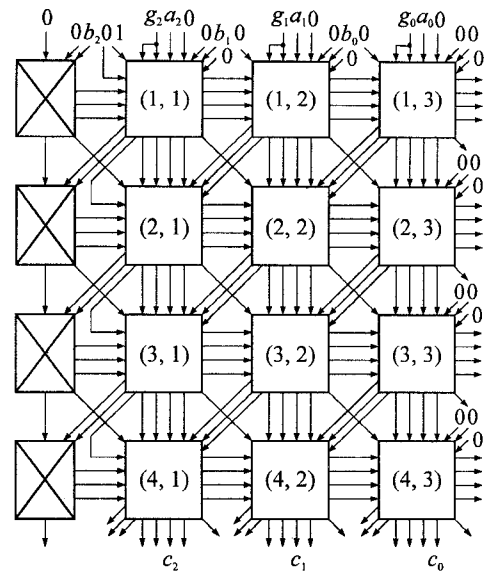

Fig. 4. A new variant of the DG shown in Fig. 1.

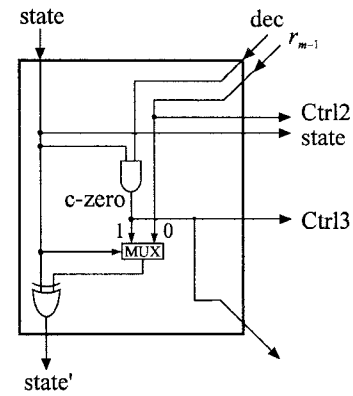

Fig. 5. The circuit of Type-3 cell in Fig. 4.

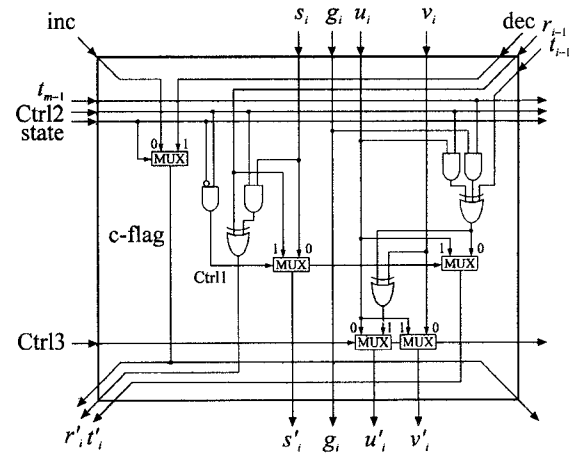

Fig. 6. The circuit of Type-4 cell in Fig. 5.

and "•" denotes a 1-bit 1-cycle delay element. The SFG array is controlled by a sequence 011···1 of length m. According to the projection, each basic cell of the SFG array should contain the circuitry of Type-3 and Type-4 cells. Since the control signals Ctrl2 and Ctrl3, state, and t_{m-1} of the ith iteration must be broadcast to all the Type-4 cells in the ith row of the DG,

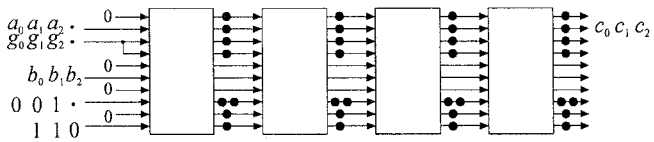

Fig. 7. A one-dimensional SFG array for division in $GF(2^3)$.

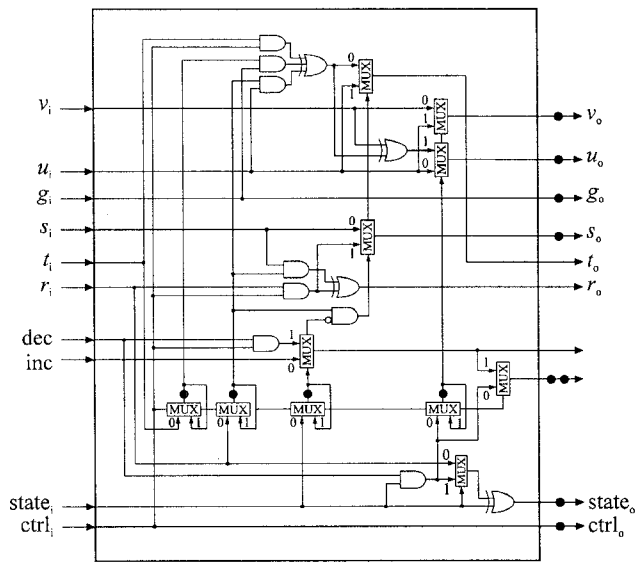

Fig. 8. The circuit of the basic cell in Fig. 7.

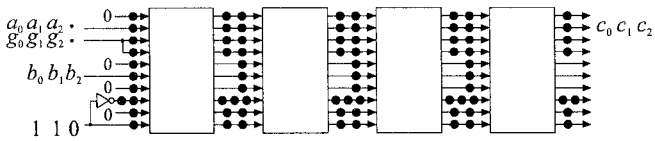

Fig. 9. A new serial-in serial-out systolic array for division in $GF(2^3)$.

extra four 2-to-1 multiplexers and four 1-bit latches are added to each basic cell of the SFG array for this purpose. Three 2-input AND gates are also added to each basic cell of the SFG array owing to the fact that three zeros must be fed to each row of the DG from the rightmost cell. When the control signal is in logic 0, these AND gates generate three zeros. Besides, each basic cell of the SFG array incorporates one 2-to-1 multiplexer for data flow arrangement.

The SFG array in Fig. 7 can be easily retimed by using the cut-set systolization techniques [16] to derive a serial-in serial-out systolic array as shown in Fig. 9. When the input data come in continuously, this array can produce output results at a rate of one per m cycles after an initial delay of $5m - 4$ cycles. The coefficients of division result $C(x)$ will emerge from the right-hand side of the array in serial form with the MSB first.

From Section II, it can be seen that the operation "$T = x \cdot T \bmod G$" of the division algorithm can be simplified to "$T = x \cdot T$" while computing inversion in $GF(2^m)$. Thus, the systolic array in Fig. 9 can also be simplified for computing inversion. For performing "$T = x \cdot T$", t_{m-1} and g_i's ($0 \le i \le m - 1$) are no more needed and thus the logic gates associated with them can be saved. Figs. 10 and 11 show the correspondingly

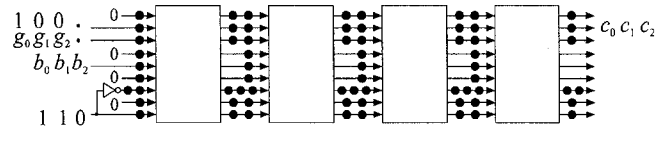

Fig. 10. A new serial-in serial-out systolic array for inversion in $GF(2^3)$.

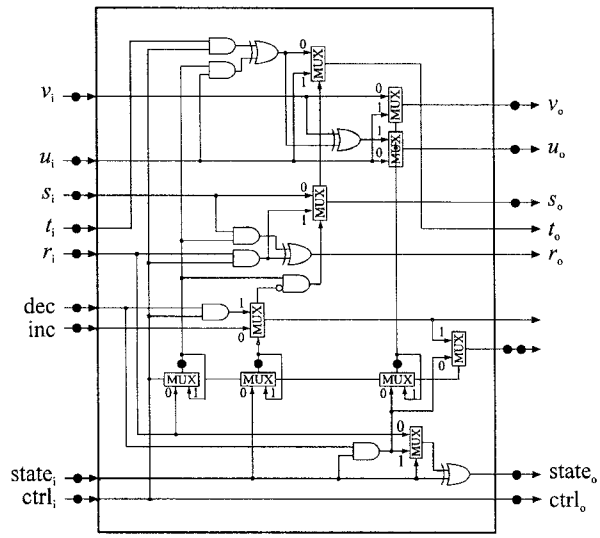

Fig. 11. The circuit of the basic cell in Fig. 10.

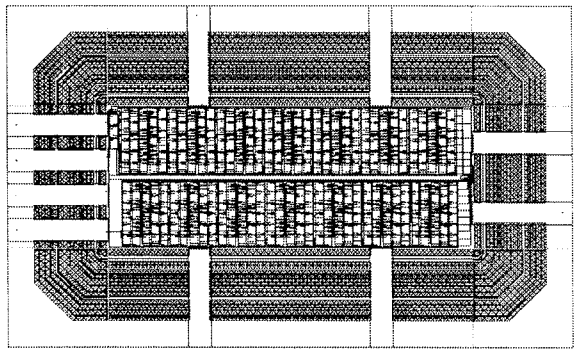

Fig. 12. A chip layout of the proposed systolic array for division in $GF(2^8)$.

simplified systolic array and basic cell, respectively. Each basic cell saves one 2-to-1 multiplexer, one 2-input AND gate, and three latches.

D. Chip Implementation

For the purpose of verification, a prototype chip of the proposed systolic array for division in $GF(2^8)$ was designed based on the COMPASS 0.6 μm CMOS standard cell library. We completed the work by using the VERILOG hardware description language and the CADENCE CAD tool. The resulting chip layout is shown in Fig. 12, where the entire die size is about 1.66×2.73 mm^2 and the core area is about 0.67×1.62 mm^2. The critical-path delay of the core circuitry is around 3 ns under the technology used. When the I/O pad delay is considered, it was estimated that the chip can run at a clock rate up to 167 MHz.

IV. Conclusions

In this paper, two serial-in serial-out systolic arrays have been presented for computing inverses or divisions in $GF(2^m)$ over the standard basis. One is for division and the other one is for inversion. Both proposed arrays possess the features of regularity, modularity, and unidirectional data flow. Thus, they are well suited to be implemented using VLSI techniques with fault-tolerant design [13]-[14].

Table I gives a comparison of the proposed arrays with existing related systolic designs. All the circuits compared have the same time-complexity of $O(m)$ and I/O format of serial-input serial-output. From this table, we can see that our proposed arrays have the smallest latency among the circuits compared and the same throughput performance as the circuits in [10]-[11]. Besides, our circuits involve only $O(m)$ area-complexity, which is a significant improvement in hardware complexity, and one control signal, instead of two for the circuits in [9] and [11]. Table II gives transistor count estimations of the circuits compared in Table I based on the following assumptions: an inverter, a 2-input AND gate, a 2-input XOR gate, a 2-to-1 multiplexer, a 2-input OR gate, and a 1-bit latch consist of 2, 4, 6, 6, 6, and 8 transistors, respectively [17]. According to the estimations, our proposed arrays have less transistor counts over a wide range of m and have substantial reductions in transistor counts when the value of m is large. For example, considering the case of $m = 1000$, the transistor counts of the circuits in Figs. 9 and 10 are only about 1.89% and 1.64% of that of the circuits in [10]-[11], respectively. Therefore, our proposed arrays are well suited to various applications, especially for those applications with large value of m, such as cryptography.

References

[1] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. Cambridge, MA: MIT Press, 1972.

[2] E. R. Berlekamp, *Algebraic Coding Theory*. New York: Mcgraw-Hill, 1968.

[3] D. E. R. Denning, *Cryptography and Data Security*. Reading, MA: Addsion-Wesley, 1983.

[4] M. Kovac, N. Ranganathan and M. Varanasi, "SIGMA: A VLSI systolic array implementation of a Galois field $GF(2^m)$ based multiplication and division algorithm," *IEEE Trans. VLSI Systems*, vol. 1, pp. 22-30, Mar. 1993.

[5] S.-W. Wei, "VLSI architectures for computing exponentiations, multiplicative inverses, and divisions in $GF(2^m)$," in *Proc. 1995 IEEE Int. Symp. Circuits Syst.*, London, May 1995, pp. 4.203-4.206.

[6] C.-L. Wang and J.-H. Guo, "New systolic arrays for $C + AB^2$, inversion, and division in $GF(2^m)$," in *Proc. 1995 European Conference Circuit Theory Design*, Istanbul, Turkey, Aug. 1995, pp. 431-434.

[7] J.-H. Guo and C.-L. Wang, "Systolic array implementation of Euclid's algorithm for inversion and division in $GF(2^m)$," in *Proc. 1996 IEEE Int. Symp. Circuits Syst.*, Atlanta, May 1996, pp. II.481-II.484.

[8] J.-H. Guo and C.-L. Wang, "Hardware-efficient systolic array implementations of Euclid's algorithm for inversion and division in $GF(2^m)$," in *Proc. 1996 Int. Comput. Symp.–Int. Conf. Comput. Architecture*, Kaohsiung, Taiwan, Dec. 1996, pp. 221-228.

[9] C.-L. Wang and J.-L. Lin, "A systolic architecture for computing inverses and divisions in finite fields $GF(2^m)$," *IEEE Trans. Comput.*, vol. 42, pp. 1141-1146, Sep. 1993.

[10] M. A. Hasan and V. K. Bhargava, " Bit-level systolic divider and multiplier for finite fields $GF(2^m)$," *IEEE Trans. Comput.*, vol. 41, pp. 972-980, Aug. 1992.

[11] S. T. J. Fenn, M. Benaissa, and D. Taylor, "$GF(2^m)$ multiplication and division over the dual basis", *IEEE Trans. Comput.*, vol. 45, pp. 319-327, Mar. 1996.

[12] H. T. Kung, "Why systolic architectures?," *Computer*, vol. 15, pp. 37-46, Jan. 1982.

[13] H. T. Kung and M. Lam, "Fault tolerant and two level pipelining in VLSI systolic arrays," in *Proc. MIT Conf. Advanced Res. VLSI*, Cambridge, MA, Jan. 1984, pp. 74-83.

[14] J. V. McCanny, R. A. Evans, and J. G. McWhirter, "Use of unidirectional data flow in bit-level systolic array chips," *Electron. Lett.*, vol. 22, pp. 540-541, May 1986.

[15] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[16] S. Y. Kung, "One supercomputing with systolic/wavefront array processors," *Proc. IEEE*, vol. 72, pp. 867-884, July 1984.

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

Table I
Comparison of some systolic arrays for inversion/division in $GF(2^m)$ with the same time complexity and I/O format

Item \ Circuits	[9]-[11]	Figs. 9 and 10
Throughput (1/cycles)	$1/(2m-1)$ [9] $1/m$ [10], [11]	$1/m$
Latency (cycles)	$7m-3$ [9] $5m-1$ [10], [11]	$5m-4$
Area complexity	$O(m^2)$	$O(m)$
Area-time product	$O(m^3)$	$O(m^2)$
# of control signals	2 [9], [11] 1 [10]	1
Operation	Division	Division Fig. 9 Inversion Fig. 10

Table II
Transistor count estimations of the circuits listed in Table I

Circuits \ m	8	12	16	32	50	80	100	300	500	1000
[10] & [11]	3458	6762	11090	38642	89.2k	219.6k	338.5k	2936k	8093k	32186k
Fig. 9	4256	6688	9120	18848	29.8k	48k	60.2k	181.8k	303.4k	607.4k
Fig. 10	3696	5808	7920	16368	25.9k	41.7k	52.3k	157.9k	263.5k	527.5k

* The transistor count of the circuit in [9] is about 3 times of that of the circuit in [10].