NOVEL DIGIT-SERIAL SYSTOLIC ARRAY IMPLEMENTATION OF EUCLID'S ALGORITHM FOR 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

In this paper, a novel digit-serial systolic array for computing divisions in $GF(2^m)$ over the standard basis is presented. To the authors' knowledge, this is the very first digit-serial systolic divider for $GF(2^m)$. The proposed architecture possesses the features of regularity, modularity, and unidirectional data flow. Thus, it is well suited to be implemented using VLSI techniques with fault-tolerant design. One important feature of the proposed architecture is that different throughput performances can be easily achieved by varying the digit size. By choosing the digit size appropriately, the proposed digit-serial architecture can meet the throughput requirement of a certain application with minimum hardware.

1. INTRODUCTION

Finite fields $GF(2^m)$ have played an important role in areas of communications. Some applications, such as error-correcting codes [1]-[2] and cryptography [3], usually involve the division operation in finite fields $GF(2^m)$. Performing such arithmetic using software on a general-purpose computer is a straightforward method, but it will be neither fast enough nor cost effective for related real-time applications, especially for the public-key cryptosystems where large fields are adopted [3]. Therefore, special-purpose architectures for $GF(2^m)$ division become indispensable.

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]-[12] are designed based on the concepts of systolic architecture [13]. Existing architectures for inversion and/or division in $GF(2^m)$ can be categorized into two types: bit-parallel and bit-serial architectures. Basically, a bit-parallel system reaches much better throughput performance than a bit-serial one, but it involves much more circuit complexity. For some applications, bit-serial computation may be too slow and fully bit-parallel computation may be faster than necessary and too hardware-consuming. To improve the trade-off between throughput performance and hardware complexity, the adoption of digit-serial architectures [14]-[15] seems to be a good approach.

In this paper, a novel digit-serial systolic divider for $GF(2^m)$ over the standard basis is presented. If the input data come in continuously, it can produce division results at a rate of one every m / L clock cycles with a latency of $(5m / L) - 1$ clock cycles, where L is the selected digit size. The proposed array is highly regular and modular and thus well suited to VLSI implementation. As we will see that the proposed divider and some existing dividers reach the smallest Area-Time (AT) product of $O(m^2)$. The most important feature of the proposed architecture is that different throughput performances can be easily achieved simply by varying the digit size. If the digit size is chosen appropriately, the proposed digit-serial architecture can meet the throughput requirement of a certain application with minimum hardware.

2. THE DIVISION ALGORITHM FOR $GF(2^m)$ IN [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 division $A(x) / B(x) \bmod G(x)$, the algorithm in [8] based on the Euclid's algorithm can be used. It consists of $2m$ iterations and can be summarized as follows:

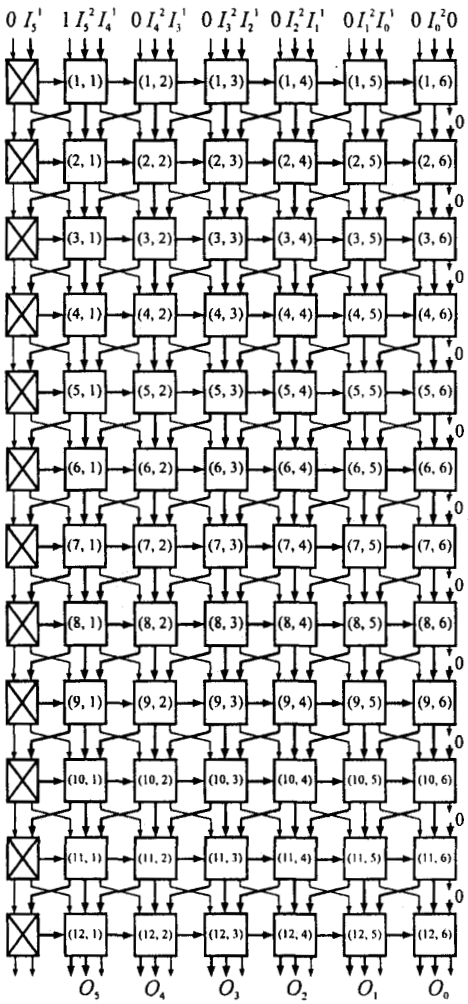
The $GF(2^m)$ division algorithm in [8]
$R = B(x)$; $S = G = G(x)$; $U = A(x)$; $V = T = 0$;
state = 0; count = 0;
for $i = 1$ to $2m$ **do**
  $R = x \cdot R$; $T = x \cdot T \bmod G$;
  if state = = 0 **then**
    count = count + 1;
    if r_m = = 1 **then** (* r_m: coefficient of x^m of R *)
      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 (* V has the result $C(x)$ *)

After $2m$ iterations, polynomial V contains the division result $C(x)$. Two parallel-in parallel-out architectures and a serial-in serial-out architecture were developed based on this algorithm in [8] and [12]. In the following, a digit-serial architecture is given for the same problem.

3. NOVEL DIGIT-SERIAL SYSTOLIC ARRAY IMPLEMENTATION

3.1 Dependence graph of the above division algorithm [12]

0-7803-4455-3/98/$10.00 © 1998 IEEE



Input: I_i^1 : [0, b_i, 0] I_i^2 : [g_i, g_i, a_i, 0]

Output: O_i : [*, g_i, *, c_i] *: don't care

Fig. 1. A dependence graph for division in $GF(2^6)$.

Fig. 1 shows a dependence graph (DG) of the above division algorithm for $GF(2^m)$, where $m = 6$. It consists of $2m$ Type-1 cells and $2m \times m$ Type-2 cells, where the functions of these two types of basic cells are depicted in Figs. 2-3. The coefficients of $A(x)$, $B(x)$, and $G(x)$ enter the array from the top, and the cells in the i-th row of this array realizes the i-th iteration of the algorithm. The value of count is traced based on the tracing scheme in [12], where each Type-2 cell incorporates one 2-to-1 multiplexer (the one with Inc and Dec as its inputs and C-flag as its output) for this purpose. The value of count will increase or decrease depending on the value of state. That is

$$\text{count}' = \begin{cases} \text{count} + 1, & \text{if state} == 0 \\ \text{count} - 1, & \text{if state} == 1 \end{cases} \quad (5)$$

where count' represents the value of count for the next iteration. Besides, the logic value of C-zero of Type-1 cell can be used to determine whether the value of count equals to zero. For the Type-1 in the i-th row, "C-zero = 1" means that "count = 0" after i iteration, while "C-zero = 0" means that "count ≠ 0" after i iteration (for more details, see [12]).

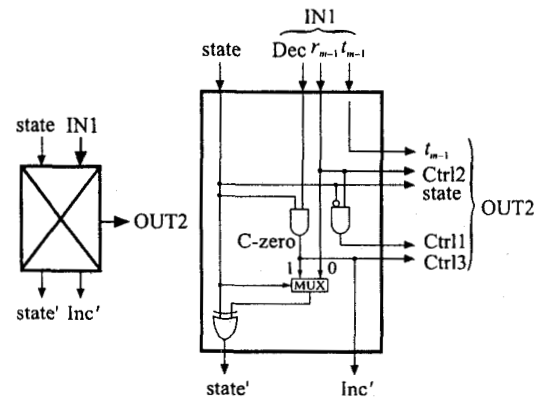


Fig. 2. The circuit of Type-1 cell in Fig .1.

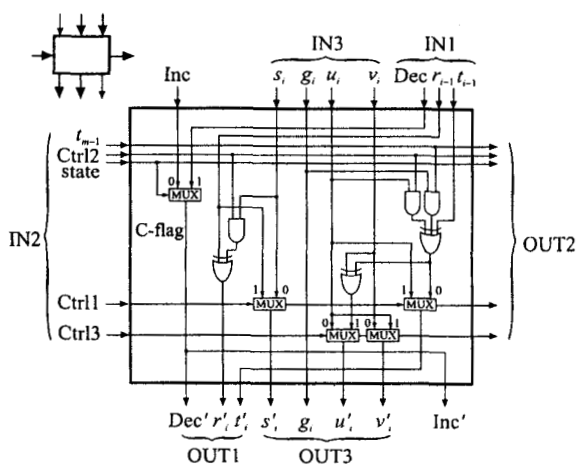


Fig. 3. The circuit of Type-2 cell in Fig. 1.

For each row, Type-1 cell also generates the control signals Ctrl1, Ctrl2, and Ctrl3 for the present iteration as well as computes the value of state for the next iteration (i.e., state' in Fig. 2) according to the following logic functions:

$$\text{Ctrl1} = (\text{state} == 0) \,\&\, (r_m == 1) \quad (6)$$

$$\text{Ctrl2} = (r_m == 1) \quad (7)$$

$$\text{Ctrl3} = (\text{C-zero} == 1) \quad (8)$$

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

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 the division result $C(x)$ will emerge from the bottom of the array (i.e., after $2m$ iterations).

3.2 Novel digit-serial systolic divider for $GF(2^m)$

To design a digit-serial systolic divider for $GF(2^m)$ with digit-size L $(L \ge 2)$, we can proceed according to the following procedure:

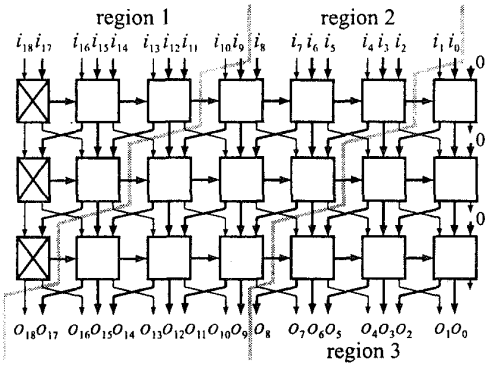


Fig. 4. Partition one part of the DG in Fig. 1 with $L = 3$.

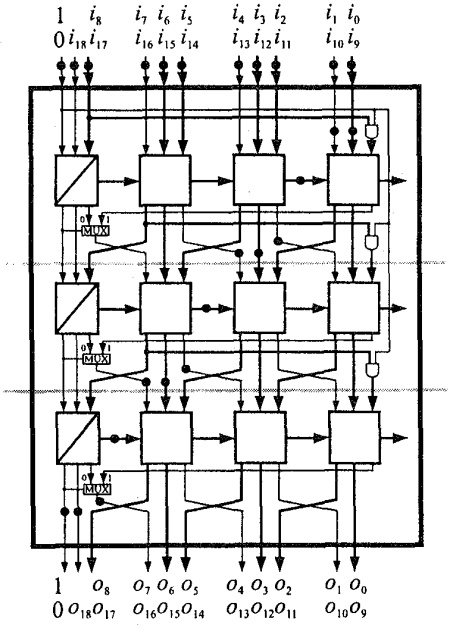


Fig. 5. Implement the circuit in Fig. 4 using one cell with $L = 3$.

1) Partition the DG in Fig. 1 into $2m / L$ identical parts, where each part consists of L rows. For example, we can partition the DG in Fig. 1 into 4 parts with $L = 3$.
2) Partition one part of the DG into $(m / L) + 1$ regions. Fig. 4 shows an example of $L = 3$. Region 1 and region $(m / L) + 1$ are in triangular forms while the other regions are in the form of parallelogram. Region 1 contains L Type-1 cells and $(L - 1)L / 2$ Type-2 cells, region i $(2 \le i \le m / L)$ comprises of L^2 Type-2 cells, and region $(m / L) + 1$ includes $(L + 1)L / 2$ Type-2 cells.
3) Scheduling: Those regions in Fig. 4 are defined as *equitemporal regions*, all the processor elements belonging to the same region are processed at the same time. The processor elements in region i are scheduled to be processed at the i-th clock cycle.
4) To implement the circuit in Fig. 4 according to the above schedule, the cell in Fig. 5 can be used, where "•" denotes a one-cycle delay element. The cell is composed of L Type-3 cells, shown in Fig. 6, L^2 Type-2 cells, $3L$ 2-input AND gates, L 2-to-1 multiplexers, and $19L + 4$ one-bit one-cycle delay elements. A control sequence Ct = 011···1 of length m / L is used to control the cell. Since Ctrl1, Ctrl2, Ctrl3, state, and t_{m-1} from each Type-1 cell in Fig. 4 must be broadcast to all the Type-2 cells in the same row, each Type-3 cell incorporates four 2-to-1 multiplexers and four one-bit latches for this purpose. When the control signal Ct is in logic 0, Ctrl2, Ctrl3, state, and t_{m-1} are loaded. Besides, $3L$ 2-input AND gates are added to the cell in Fig. 5 owing to the fact that three zeros must be fed into each row of the circuit in Fig. 4 from the rightmost cell. When the control signal is in logic 0, these AND gates generate zeros. In addition to above, the cell in Fig. 5 also incorporates L 2-to-1 multiplexers for data flow arrangement.
5) By concatenating $2m / L$ cells shown in Fig. 5 together, a digit-serial systolic divider for $GF(2^m)$ can be obtained. Fig. 7 shows the result. The coefficients of $A(x)$, $B(x)$, and $G(x)$ enter the array from the left at a rate of L-bit every clock cycle and those of $C(x)$ emerge from the right of the array at the same rate with a latency of $(5m / L) - 1$ clock cycles. That is, the systolic divider can produce division results at a rate of one every m / L clock cycles.

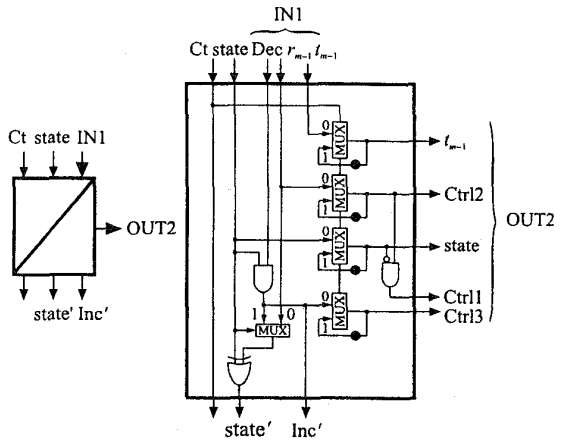


Fig. 6. The circuit of Type-3 cell in Fig. 5.

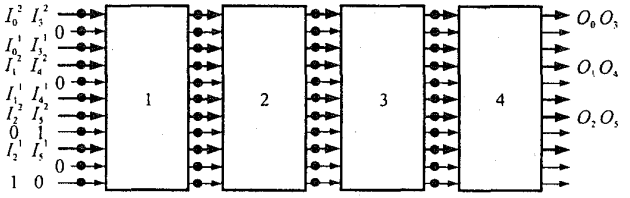


Fig. 7 A novel digit-serial systolic divider for $GF(2^6)$ with digit size $L = 3$.

When the digit size L gets large, the maximum propagation delay of the cell in Fig. 5 will become large and thus the clock rate will decrease. To conquer such a problem, we can further pipeline the cell in Fig. 5 so that the maximum propagation delay can be kept small when the digit size L gets large. For example, we can further pipeline the cell in Fig. 5 into 3 stages by placing one one-cycle delay element on each of the communication links crossed by the gray lines.

4. CONCLUSIONS

In this paper, a novel digit-serial systolic divider for $GF(2^m)$ over the standard basis is presented. To the authors' knowledge, this is the very first digit-serial systolic divider for $GF(2^m)$. The proposed divider possesses the features of regularity, modularity, and unidirectional data flow. Thus, it is well suited to be implemented using VLSI techniques with fault-tolerant design [16]-[17].

TABLE I gives a comparison of some systolic arrays for division in $GF(2^m)$. The systolic divider in [4] is not considered for comparison due to its large area-complexity of $O(m * 2^m)$. It can be seen that the proposed divider and the dividers in [7]-[8] and [12] reach the smallest Area-Time (AT) product of $O(m^2)$. As compared to the dividers in [9]-[11], the proposed systolic divider has better performances since it involves less area-complexity and reaches higher throughput rate. For the dividers in [5]-[7] and Fig. 4 of [8], they are only suitable for some very high-speed applications where the value of m is not large. For those applications where large value of m is adopted, their large area-complexity will make single-chip implementation impossible. As for the divider in Fig. 8 of [8] and the divider in [12], they involve small area-complexities but their low throughput rates may be too slow for some real-time applications. The proposed digit-serial systolic divider has a throughput rate of L / m and involves $O(Lm)$ area-complexity. By varying the digit size L, different throughput rates can be easily achieved. For a certain application, throughput requirement can be met by choosing the digit size L appropriately. By choosing the digit size L appropriately, the proposed digit-serial architecture can meet the throughput requirement of a certain application with minimum hardware.

5. 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, Sept. 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] J.-H. Guo and C.-L. Wang, "Bit-serial systolic array implementation of Euclid's algorithm for inversion and division in $GF(2^m)$," in *Proc. 1997 Int. Symp. on VLSI Technology, Systems, and Applications*, Taipei, Taiwan, June 1997, pp. 113–117.

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

[14] R. L. Hartley and P. F. Corbett, "Digit-serial processing techniques," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 707–719, June 1990.

[15] R. L. Hartley and P. F. Corbett, "Designing systolic arrays using digit-serial arithmetic," *IEEE Trans. Circuits Syst.-II: Analog and Digital Signal Processing*, vol. 39, pp. 62–65, Jan. 1992.

[16] 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.

[17] 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.

TABLE I COMPARISON OF SOME SYSTOLIC ARRAYS FOR DIVISION IN $GF(2^m)$

Item \ Circuits	The circuits in [5] & [6]	The circuits in [7] & [8]	The circuits in [9]-[12]	The proposed divider
Throughput rate (unit=1/cycles)	1	**[7] & Fig. 4 in [8]** 1 **Fig. 8 in [8]** $1 / (2m - 2)$	**[9]** $1 / (2m - 1)$ **[10]-[12]** $1 / m$	L / m
Time complexity	$O(1)$	**[7] & Fig. 4 in [8]** $O(1)$ **Fig. 8 in [8]** $O(m)$	$O(m)$	$O(m / L)$
Area complexity	$O(m^3)$	**[7] & Fig. 4 in [8]** $O(m^2)$ **Fig. 8 in [8]** $O(m)$	**[9]-[11]** $O(m^2)$ **[12]** $O(m)$	$O(Lm)$
Latency (unit=cycles)	$O(m^2)$	$O(m)$	$O(m)$	$O(m / L)$
I/O format	parallel-in parallel-out	parallel-in parallel-out	serial-in serial-out	digit-serial
Area-Time product	$O(m^3)$	$O(m^2)$	**[9]-[11]** $O(m^3)$ **[12]** $O(m^2)$	$O(m^2)$

* L is the digit size.