High-Speed C-Testable Systolic Array Design for Galois-Field Inversion

Chih-Tsun Huang and Cheng-Wen Wu*

Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan

Abstract

Systolic architectures for inversion in Galois field ($GF(2^m)$) are presented. The proposed inversion algorithm is a counter-free extended Euclidean algorithm, which results in simple circuit implementation for GF inversion. Additionally, the bit-parallel implementation proposed is shown to be C-testable. Testability and modularity make it suited to VLSI implementation.

1 Introduction

Arithmetic in finite field (or Galois field, $GF(2^m)$) has found a wide-spread application in error control coding, switching theory, cryptography, etc. Several division or inversion design have been proposed [1–3], but most of them are not suitable for VLSI implementation due to complex routing, nonmodular architectures, and low testability. Testability especially is an increasing concern in VLSI design.

In 1993, Brunner *et al.* [3] presented a modular inverse structure based on Euclidean algorithm, which has global control signals and are not pipelinable. Moreover, modification is necessary for changing the irreducible polynomial. An improvement was recently proposed in [4], showing a bit-parallel systolic array implementation based on the Euclidean algorithm for both inversion and division.

In this paper, systolic bit-parallel architectures for inversion in $GF(2^m)$ are proposed. Our inverse algorithm is based on a novel improvement in the extended Euclidean algorithm. The proposed algorithm results in a simple, regular, and testable structure which is highly suitable for VLSI implementation. Moreover, it is independent of the irreducible polynomial selected. The proposed algorithm is a true counter-free extended Euclidean algorithm without degree counting or zero checking which the systolic structure of [4] suffers. With a proper initialization and an extra α^{-m} multiplier, our modified extended Euclidean algorithm can compute the $GF(2^m)$ division. We also present systolic-array implementations, including the bit-parallel array for high throughput and the modified bit-parallel array for high testability.

*This work was supported in part by the National Science Council, R.O.C., under Contract NSC85-2215-E007-014.

2 Modified Extended Euclidean Algorithm

The Extended Euclidean Algorithm (XEA) addressed is basically the one that is used in Reed-Solomon decoding [5]. In this section, we will present a modified XEA (MXEA) and its VLSI implementation.

There are problems in the XEA. One of the most concerned is that it requires a degree comparison between the two polynomials in each iteration. In [6], each systolic stage has a counter to perform the comparison. This requirement leads to costly cell design. Moreover, the cell area grows with respect to m. In [3], the counter is separated from the modular cells, avoiding a complex structure for the individual cells. This however results in a nonsystolic architecture. Furthermore, four global signals make it unattractive for VLSI implementation.

We propose the following modified recursive operation for GCD on the two polynomials, $A(x)$ and $F(x)$, which is different from XEA.

$$\begin{aligned} F &= q_0 A + R_0, \\ q_0 A &= q_1 R_0 + R_1, \\ q_1 R_0 &= q_2 R_1 + R_2, \\ &\vdots \\ q_{k-1} R_{k-2} &= q_k R_{k-1} + R_k, \end{aligned} \tag{1}$$

where $\forall i = 0, 1, \ldots, k-1$, $q_i \in \{x, x^2, \ldots, x^m\}$, $q_k = 1$, $R_k = 0$, and $q_k R_{k-1} = GCD(x^m A, x^m F)$. Eq. (1) guarantees $q_{k-1} R_{k-2} = q_k R_{k-1} = x^m$. The reason follows. Begin with the polynomials F and A, where $\deg(F) = m$ and $\deg(A) \leq m - 1$. Since

ED&TC '97 on CD-ROM

$\deg(q_0 A) = m$, $\deg(R_0) \leq m-1$. As a result, $q_1 R_0$ has at least one more trailing zero as compared with F (i.e., $\deg(q_1 R_0) = m$). By induction, it is guaranteed that $q_i R_{i-1}$ has at least one more trailing zero as compared with $q_{i-1} R_{i-2}$. Also, $\forall i = 0, 1, \ldots, k$, $\deg(q_{i-1} R_{i-2}) = \deg(q_i R_{i-1}) = m$. Therefore, both $q_k R_{k-1}$ and $q_{k-1} R_{k-2}$ will equal x^m within $k \leq 2m$ iterations due to the $2m$ trailing terms of F and A.

$$\begin{aligned} GCD(F, A) &= GCD(A, R_0), \\ GCD(q_0 A, R_0) &= GCD(R_0, R_1), \\ GCD(q_1 R_0, R_1) &= GCD(R_1, R_2), \\ &\vdots \\ GCD(q_{k-2} R_{k-3}, R_{k-2}) &= GCD(R_{k-2}, R_{k-1}). \end{aligned} \tag{2}$$

Because $GCD(A, R_0) = 1$, $GCD(q_0 A, R_0) = x^i \leq q_0 GCD(A, R_0) = q_0$, i.e., $\deg(x^i) \leq \deg(q_0)$,

$$\begin{aligned} & GCD(q_{i-1} R_{i-2}, R_{i-1}) = x^j \\ \leq\ & q_{i-1} GCD(R_{i-2}, R_{i-1}) \\ & = q_{i-1} GCD(q_{i-2} R_{i-3}, R_{i-2}) \\ \leq\ & q_{i-1} q_{i-2} GCD(R_{i-3}, R_{i-2}) \\ & = q_{i-1} q_{i-2} GCD(q_{i-3} R_{i-4}, R_{i-3}) \\ & \vdots \\ \leq\ & q_{i-1} q_{i-2} \cdots q_1 GCD(R_0, R_1) \\ & = q_{i-1} q_{i-2} \cdots q_1 GCD(q_0 A, R_0) \\ \leq\ & q_{i-1} q_{i-2} \cdots q_0 GCD(A, R_0) \\ & = q_{i-1} q_{i-2} \cdots q_0 GCD(F, A), \end{aligned} \tag{3}$$

where $j \leq \deg(q_{i-1} q_{i-2} \cdots q_0)$. Consequently, $GCD(q_{k-1} R_{k-2}, R_{k-1}) = x^l GCD(F, A)$, $l \leq \deg(q_{k-1} q_{k-2} \cdots q_0)$, and $R_{k-1} = q_k R_{k-1} = GCD(x^m F(x), x^m A(x))$ because $q_k R_{k-1} = x^m$ and $l = m$. Eventually, the result R_{k-1} will be $GCD(x^m A, x^m F)$ instead of $GCD(A, F)$, i.e., this algorithm computes $GCD(x^m A, x^m F) = x^m$ successfully. The MXEA is summarized below.

```
MODIFIED-EXTENDED-EUCLIDEAN-ALGORITHM()
 1  P_0 := F(x); V_0 := 0;
 2  R_0 := A(x); U_0 := α^{-m};
 3  i := 0;
 4  repeat
 5        i := i + 1;
 6        if r_m = 0
 7          then
 8                R_i := x · R_{i-1};  (* deg(R_{i-1}) < m *)
 9                U_i := x · U_{i-1} mod F;
10                  (* ensures that U ∈ GF(2^m) *)
11          else  (* r_m = 1 *)
12                R_i := x · (P_{i-1} − R_{i-1});
13                  (* deg(R_{i-1}) = deg(P_{i-1}) = m *)
14                U_i := x · (V_{i-1} − U_{i-1}) mod F;
15                P_i := R_{i-1};
16                V_i := U_{i-1};  (* V, U perform the
17                      same operation as P, R *)
18    until (P_i = x^m and R_i = x^m)
19  return U_i;  (* B(x) = V_i = U_i *)
```

Because $x^m A$ is simply m left shifts of $A(x)$, the result is exactly the same as that for the original XEA after the shifting, i.e., $U = \alpha^m A^{-1}$ if initially $U = 1$ and $V = 0$ and we apply the XEA to the polynomial pairs $\{(R, U), (P, V)\}$. The extra α^m factor can be eliminated if initially $U = \alpha^{-m}$. The recursive operation of (1) guarantees that the higher degree of the two polynomials is fixed to m during the computation. Hence, the degree comparison of the two polynomials is simply the comparison of the coefficients of their mth terms.

The proposed MXEA needs no degree counter, which is a general requirement for the original XEA. Since MXEA is *counter-free*, there is no routing irregularity due to the increase of m. In [4], however, area and time complexity grows with respect to m due to the degree counter and zero checker in the boundary cell.

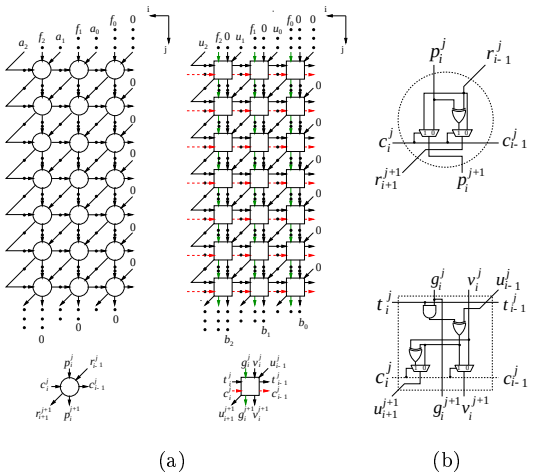
Our MXEA can be turned into a division algorithm in a similar way as that given in [4]. If we let $R_0 = A$ and $U_0 = C$ in the beginning, and apply the MXEA shown above, then the final result is $U_i = \alpha^m A^{-1} C$. In [4], an α^{-m} scaler (a special-purpose multiplier) is used to correct this extra factor. The same approach can be applied to our MXEA for division.

The time complexity of the original XEA is $O(2m)$. our MXEA does not guarantee the linear time complexity. The number of iterations in (1) is $O(m)$, but in each iteration, $q_i \in \{x, x^2, \ldots, x^m\}$, i.e., in the worst case it is necessary to perform m shifts in one iteration. As a result, the asymptotic bound is $O(m^2)$ both in time and area.

3 VLSI Implementation

3.1 Bit-Parallel Architecture

Without loss of generality, the bit-parallel systolic architecture for $GF(2^3)$ derived from the MXEA is shown in Fig. 1(a). An implementation of the MXEA can be represented by two main blocks: one performs the GCD evaluation of F and A, and the other computes the inverse of A simultaneously. In Fig. 1(a), the systolic array on the left is the GCD evaluation block. Since the mth bit of F is always 1, it is removed for simplicity. Two m-tuple inputs, F and A, are entered from the top row in parallel. As a result, the GCD, which is always x^m (or the $(m+1)$-tuple



(a) (b)

Figure 1: (a) Bit-parallel systolic array. (b) Schematic of the basic cells.

$(10\cdots0)$), will be produced from the bottom row in parallel. Again, the leading 1 is transparent in this design. Since the algorithm always performs single shifts of R and U in the first stage, respectively (because the degree of A must be less than that of F), the shifting is done by the diagonal connections on the top of the array, and the first stage can be omitted in practice. Similarly, the inverse will be produced at the bottom of the inverse computation array at the same time as the GCD (x^m) is produced.

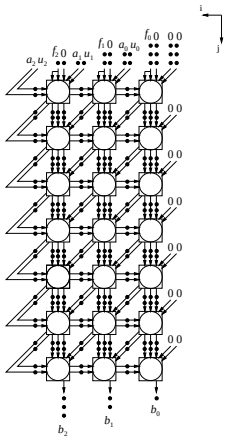


Figure 2: Structure of the proposed systolic array.

Fig. 1(b) shows the schematic of the basic cells. The circular cell represents the basic cell for GCD evaluation, while the square cells are for inverse computation. Note that these two different cells should be merged into one for the same control signal c as illustrated in Fig. 2. The simple cell design results in a reduction of the chip area.

The proposed circuit structure shown in Fig. 1(b) is much simpler than that presented in [4], with only about a half of the gate count. Our design also requires less interconnection (about 50%) for each cell, resulting in a great reduction of pipeline latches. Moreover, no extra boundary cell is needed, which retains the regularity of the array. The boundary cell with degree counter and zero checker in [4] causes routing problem when m increases, since the area complexity of the boundary cell is $O(\log_2(m+1))$. Although the claimed pipeline throughput is 1 in [4], it is true only when the counter has a constant counting time (independent of m), which obviously cannot be achieved. The real time and area complexity should be multiplied by $\log_2(m+1)$. Fig. 3 illustrates the comparison of AP between these two design. For lower throughput application, a low-cost bit-serial array can be also derived from our bit-parallel one by direct projection.

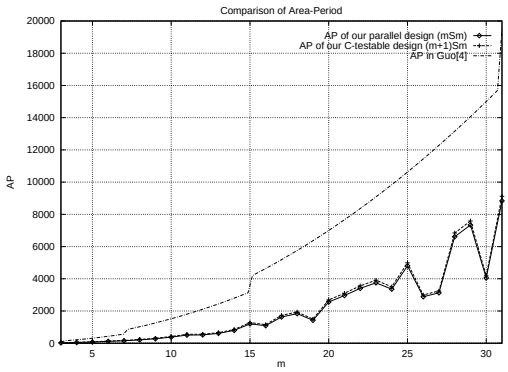


Figure 3: Comparisons of AP product.

3.2 C-Testable Bit-Parallel Architecture

Testability is more and more important to VLSI circuits as the chip size and density and its component-to-pin ratio keeps growing. Systolic arrays are usually easier to test due to their simple and regular structure. Previous works [7, 8] have identified certain conditions for C-testable systolic arrays. Our systolic GF inverse circuit is not C-testable due to the inhomogeneous connection of the left boundary of both the GCD evaluation array and the inverse computation array. An alternative bit-parallel inverter which is easily testable is presented here by a slight modification on the systolic array shown above.

We note that the functions of the basic cells in Fig. 1(a) are *bijective*, which implies that a different input combination results in a different output combination. It is an important feature for C-testability [8],

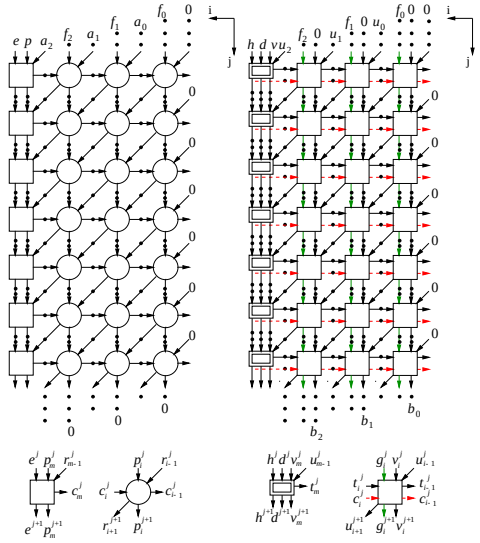


Figure 4: Easily testable systolic array for GF inverse computation.

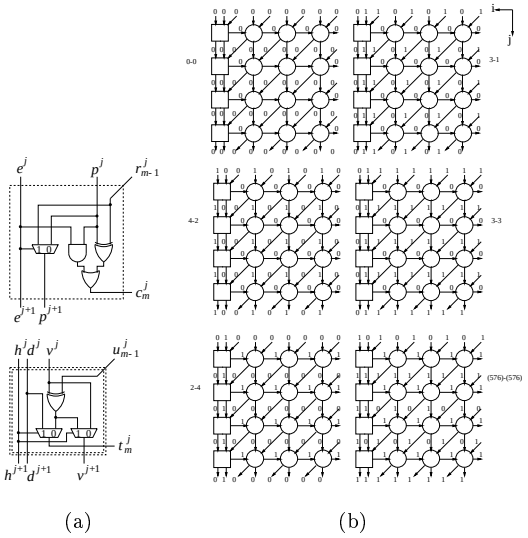


Figure 5: (a) Schematics of the boundary cells. (b) The T patterns of the GCD evaluation array.

and is applied to our testable systolic architecture.

The basic idea of our modification is to break the inhomogeneous connections of the leftmost boundary and to access these wires in test mode directly without changing the normal operation. The modified systolic architecture is shown in Fig. 4, where the boundary cells have a few extra control inputs for each inhomogeneous connection. Fig. 5(a) depicts the schematics of the boundary cells: the upper cell is the boundary cell of the GCD evaluation array, while the lower is that of the inverse computation array. For the GCD evaluation block, the systolic array operates in the normal mode when the control inputs $(e, p_m) = (0, 0)$. Other control values are to make the test patterns repeatable for all the cells. All connections between cells are local, i.e., the array remains systolic even in the test mode.

We list all the eight test patterns for the GCD evaluation array in Fig. 5(b). The eight patterns are repeatable for all the cells. Each cell receives all possible input combinations, i.e., the test set is *pseudoexhaustive* for the array (exhaustive for each cell). The property, together with the bijective cell function, guarantees that any single cell fault will be activated and the fault effect will be propagated to the primary output of the GCD evaluation array [8]. Any fault effect propagated out of the bottom of the array can be observed directly. Other faults which pass through the diagonal connections to u^j_{m-1} of the boundary cells will be propagated either to p^{j+1} or through c^j_m back to the cell array. Any fault on c^j_m will be observed on c^j_0 because c^j_i is sent to c^j_{i-1}. In summary, any single cell fault excited inside the cell array can be propagated either to the primary outputs or to p^{j+1} inside the boundary column.

Although the boundary cells are not bijective, and they do not receive pseudoexhaustive test patterns, the repeatable test patterns applied to the boundary cells still guarantee to detect all single stuck-at faults along the boundary cells on the leftmost column if the multiplexer is considered as a primitive gate. All the single stuck-at faults on the GCD evaluation array are detected by the eight patterns listed below. Note that a test pattern T is denoted as $T = (c_m)[e\ \ p_m\ \ a_{m-1}\ \ P\ \ A]$, where $P =< p_{m-1}, p_{m-2}, \cdots, p_0 >$ and $A =< a_{m-2}, a_{m-3}, \cdots, a_{-1} >$.

$$
\begin{aligned}
&T_0 = (0)\ [000 < 00\cdots00 >< 00\cdots00 >],\\
&T_1 = (0)\ [011 < 00\cdots00 >< 11\cdots11 >],\\
&T_2 = (0)\ [100 < 11\cdots11 >< 00\cdots00 >],\\
&T_3 = (0)\ [011 < 11\cdots11 >< 11\cdots11 >],\\
&T_4 = (1)\ [010 < 00\cdots00 >< 00\cdots00 >],\\
&T_{5,7,6} = (1)\ [101 < 00\cdots00 >< 11\cdots11 >],\\
&T_{7,6,5} = (1)\ [111 < 11\cdots11 >< 11\cdots11 >],\\
&T_{6,5,7} = (1)\ [110 < 11\cdots11 >< 00\cdots00 >].
\end{aligned}
$$

The symbol T_p denotes an input pattern of the array which results in a homogeneous cell input pattern whose decimal equivalent is p. The symbol $T_{p,q,r}$ denotes an input pattern of the array which results in heterogeneous cell input patterns whose decimal equivalents are p, q, and r, respectively. Furthermore,

the three patterns are tessellated cyclicly in a row-wise or column-wise manner. The signal c_m controls the operation of each row.

In our systolic design, the inverse computation array performs its normal operation while the control inputs (h, d, v_m)=(0,0,0). The way to derive the test set of the inverse computation array is similar. The basic cells of the original array receive pseudoexhaustive patterns, i.e., 32 different patterns for each cell. Furthermore, the basic cells are bijective, which guarantees that all single cell faults will be propagated either to the leftmost boundary column or to the primary outputs. The boundary cells receiving the fault from u_{m-1}^j by the diagonal connection will forward it either to t_m^j, which is routed back to the cell array, or to v^{j+1}. Contrary to the GCD boundary cells, any fault on the inputs of the boundary cell can be propagated to its outputs: v^{j+1} will be transmitted out of the bottom of the boundary column or routed back to the cell array by t_m^{j+1}. The boundary cells are not pseudoexhaustively tested, however, they are tested for all single stuck-at faults as discussed above. As a result, all single cell fault on the basic cell array as well as the single stuck-at faults on the leftmost boundary column will be detected by the 32 test patterns presented. The 32 S patterns of the inverse computation array are denoted in a similar form as the T patterns. Note that $S = (c_m)[h\ d\ v_m\ u_{m-1}\ G\ V\ U]$, where $G =< g_{m-1}, g_{m-2}, \cdots, g_0 >$, $V =< v_{m-1}, v_{m-2}, \cdots, v_0 >$, and $U =< u_{m-2}, u_{m-3}, \cdots, u_{-1} >$.

Since the two systolic blocks are combined as one physically, the two groups of test patterns can be combined into one. Only 32 test patterns as a whole are required if we apply them properly as shown below.

$$
\begin{aligned}
TS \;=\; \{&T_0S_0,\ T_1S_1,\ T_2S_2,\ T_3S_3,\ T_4S_4,\ T_{5,7,6}S_{5,7,14},\\
&T_{7,6,5}S_{7,14,5},\ T_{6,5,7}7S_{14,5,7},\ T_0S_8,\ T_1S_9,\ T_2S_{10},\\
&T_3S_{11},\ T_4S_{12},\ T_{5,7,6}S_{13,15,6},\ T_{7,6,5}S_{15,6,13},\\
&T_{6,5,7}S_{6,13,15},\ T_0S_{16},\ T_1S_{17},\ T_2S_{18},\ T_3S_{19},\ T_4S_{20},\\
&T_{5,7,6}S_{21,23,22},\ T_{7,6,5}S_{23,22,21},\ T_{6,5,7}S_{22,21,23},\\
&T_0S_{24,25},\ T_1S_{25,24},\ T_2S_{26,27},\ T_3S_{27,26},\ T_4S_{30},\\
&T_{5,7,6}S_{28,31,29},\ T_{7,6,5}S_{31,29,28},\ T_{6,5,7}S_{29,28,31}\}
\end{aligned}
$$

The control overhead is small if the test set as well as all the testing control are built in with the circuit. Only one test control and one test output are required for the built-in test design. The performance of the easily testable architecture is as good as that proposed in the previous subsection. Meanwhile, the latency and the pipeline period are the same as the original structure. The simpler cell structure leads to a great improvement on speed.

4 Conclusions

Systolic architectures which realize the MXEA have been presented. The proposed MXEA has the following advantages:

1. It eliminates degree comparison, so no degree counter is required in the GF inversion circuit.
2. It is scalable and independent of the irreducible polynomial selected.
3. With proper initial values, it can be turned into a GF divider by an extra α^{-m} multiplier [4].
4. Its high testability is suitable for VLSI implementation.

In error correcting codes such as the Reed-Solomon code, GF inversion plays a key role in the computation of the errata magnitude. A popular implementation of GF inversion is a lookup table design due to the small m in real applications. However, as m grows, the complexity of the lookup table increases in the rate of $O(2^m)$. For applications with a larger m, the proposed systolic inversion approach provides an alternative to the lookup table technique.

References

[1] 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 GF(2^m)," *IEEE Trans. Computers*, vol. 34, pp. 709–717, Aug. 1985.

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

[3] H. Brunner, A. Curiger, and M. Hofstetter, "On computing multiplicative inverses in GF(2^m)," *IEEE Trans. Computers*, vol. 42, pp. 1010–1015, Aug. 1993.

[4] J.-H. Guo and C.-L. Wang, "Systolic array implementation of Euclid's algorithm for inversion and division in $gf(2^m)$," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, (Atlanta), May 1996.

[5] T. K. Troung, W. L. Eastman, I. S. Reed, and I. S. Hsu, "Simplified procedure for correcting both errors and erasures of Reed-Solomon code using Euclidean algorithm," *IEE Proc. Pt. E*, vol. 135, no. 6, pp. 318–324, 1988.

[6] R. P. Brent and H. T. Kung, "Systolic VLSI arrays for polynomial GCD computation," *IEEE Trans. Computers*, vol. 33, pp. 731–736, Aug. 1984.

[7] A. D. Friedman, "Easily testable iterative systems," *IEEE Trans. Computers*, vol. 22, pp. 1061–1064, Dec. 1973.

[8] C.-W. Wu and P. R. Cappello, "Easily testable iterative logic arrays," *IEEE Trans. Computers*, vol. 39, pp. 640–652, May 1990.