LOW-AREA DUAL BASIS DIVIDER OVER $GF(2^M)$ *

Leilei Song *Keshab K. Parhi*

Department of Electrical Engineering
4-174 EE/CSci Building, 200 Union Street S.E.
University of Minnesota, Minneapolis, MN 55455, USA
E-mail: {llsong, parhi}@ee.umn.edu

ABSTRACT

This paper presents a low-area finite field divider using dual basis representation. This divider is based on the division algorithm of solving Discrete Wiener-Hopf Equation using Gauss-Jordan elimination method. The hardware complexity of the matrix generation part has been reduced dramatically form $O(m^2)$ to $O(m)$. When it is used as a building block for a large system, this divider can achieve more savings in hardware by utilizing sub-structure sharing techniques.

1. INTRODUCTION

Finite field arithmetic operations have received a lot of attention because of their important and practical applications in cryptography, coding theory, switching theory, and digital signal processing. Among all the arithmetic operations over finite fields, division is the most complicated operation. Some work has recently been done in this area. Two methods have been considered for finite field division. The first approach is based on the use of Fermat's theorem by recursive square and multiplication (inversion) operation [1] [2], and the second approach is based on by solving simultaneous linear equations over $GF(2)$ [3] [4]. The second division algorithm (DA) and the corresponding architecture were first completely presented in [3]. This DA consists of two steps. In the first step, a coefficient matrix is generated. In the second step, a system of linear equations is solved. When standard basis is used, the m^2 elements in coefficient matrix are all different and need to be calculated. However, when dual basis is used, only $2m-1$ different elements exist in the coefficient matrix, which form the coefficient matrix of Discrete Wiener-Hopf equation. By utilizing this particular matrix structure in dual basis, the hardware complexity of matrix generation step is dramatically reduced from $O(m^2)$ to $O(m)$.

This paper is organized as follows. In section 2, the division algorithm is briefly reviewed for both standard basis and dual basis. Section 3 presents the new dual basis divider structure and some comparison results against previous designs. Section 4 addresses the sub-structure sharing techniques which can be applied to this divider structure when it is used as a building block for large system. Finally, conclusions are drawn in section 5.

* THIS RESEARCH WAS SUPPORTED BY ARMY RESEARCH OFFICE UNDER GRANT NUMBER DA/DAAH-94-G-0405.

2. MATHEMATICAL BACKGROUND

2.1. Finite Field Fundamentals

Knowledge of basic finite field concepts and properties is assumed. They are covered in [5]- [7]. The concepts and properties related to finite field arithmetic operations are briefly reviewed in this section. In what follows, the symbols in $GF(2^m)$ and the bits of symbols, which are elements of $GF(2)$, are represented using upper and lower case variables, respectively.

Finite field $GF(2^m)$ contains 2^m elements. It is an extension field of $GF(2)$, which has elements 0 and 1. All finite fields contain a zero element, a unit element, a primitive element α and have at least one primitive irreducible polynomial $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1 x + f_0$ over $GF(2)$ associated with it where $f(\alpha) = 0$. The non-zero elements of $GF(2^m)$ can be represented as powers of the primitive element α, i.e., $GF(2^m)=\{0, \alpha^1, \alpha^2, \cdots, \alpha^{2^m-2}, \alpha^{2^m-1} = 1\}$. Since finite field $GF(2^m)$ is a vector space over $GF(2)$, its elements can also be expressed using basis representation. The *standard* basis consists of $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ and the elements of $GF(2^m)$ can be expressed using a polynomial of α as $GF(2^m)=\{A|A = a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + \cdots + a_1\alpha + a_0,$ *where* $a_i \in GF(2),\ 0 \le i \le m-1\}$. If the standard basis is taken as the primary basis, its *dual* basis can be computed as follows [7][4].

Definition 2..1 (Trace) $\forall\ \beta \in GF(p^m)$, *the trace of* β *is defined as* $Tr(\beta) = \sum_{k=0}^{m-1} \beta^{p^k}$.

Definition 2..2 (Dual Basis) *The dual basis of the primary basis* $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ *with respect to certain* $\theta \in GF(2^m)$ *is defined to be the unique set of elements* $\{\beta_0, \beta_1, \cdots, \beta_{m-1}\}$ *such that*

$$Tr(\theta\alpha^i\beta_j) = \begin{cases} 1, & when\ i = j \\ 0, & when\ i \neq j, \end{cases}$$

where θ *can be selected appropriately to simplify the conversion between standard (primary) and dual basis. In dual basis representation, The coordinates of* $A \in GF(2^m)$, a_i, *can be computed as* $a_i = Tr(\theta A\alpha^i)$, $0 \le i \le m-1$.

The main difference between standard and dual basis representations for finite field arithmetic operations is that

multiplication by α operation is performed in different ways. Given $A \in GF(2^m)$. Let $A' = A\alpha \ mod \ f(x)$, where $f(x)$ is the primitive polynomial. When standard basis representation is used for both A and A', the coordinates of A', a'_i, can be computed from the coordinates of A, a_i, as follows:

$$\begin{aligned} a'_i &= a_{i-1} + a_{m-1} f_i, \ 1 \le i \le m-1 \\ a'_0 &= a_{m-1} f_0. \end{aligned} \quad (1)$$

When dual basis representation is used for both A and A', their coordinates have the following relations:

$$\begin{aligned} a'_i &= Tr(\theta A\alpha \cdot \alpha^i) = a_{i+1}, \ 0 \le i \le m-2 \\ a'_{m-1} &= Tr(\theta A\alpha \cdot \alpha^{m-1}) = \sum_{k=0}^{m-1} a_k f_k. \end{aligned} \quad (2)$$

As can be seen, when A is multiplied by α in dual basis, only one coordinate of A', a'_{m-1}, needs to be computed. However, when standard basis representation is used, all m coordinates of A' need to be computed. This is the key fact on which the proposed designs are based.

2.2. Division Algorithm

Let $f(x)$ be the primitive irreducible polynomial of degree m for $GF(2^m)$ and let α be a root of $f(x)$. Assume $AB = C$. This multiplication can be carried out in the following way:

$$\begin{aligned} C &= AB \ mod \ f(x) \\ &= b_0 A + b_1 (A\alpha \ mod \ f(x)) \\ &\quad + b_2 (A\alpha^2 \ mod \ f(x)) + \cdots \\ &\quad + b_{m-1} (A\alpha^{m-1} \ mod \ f(x)) \end{aligned} \quad (3)$$

or in matrix form:

$$[\mathbf{A} \ \mathbf{A}\alpha \ \cdots \ \mathbf{A}\alpha^{m-1}]\mathbf{B} = \mathbf{C}, \quad (4)$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the column vectors of coordinates of A, B and C, respectively and $\mathbf{A}\ \alpha^i$ is the column vector of coordinates of $A\alpha^i \ mod \ f(x)$. When the coordinates of A and C are known, B=C/A can be computed by solving the system of m linear equations for m unknowns. This leads to the division algorithm as follows:

1. Form the coefficient matrix $\mathbf{M}$ from A, one column at a time, and the successive column is computed by multiplying its immediate previous column by α (Matrix Generation).
2. Solve (4) $\mathbf{M}\ \mathbf{B} = \mathbf{C}$ to obtain the coordinates of B by performing elementary row operations on matrix $[\ \mathbf{M}\ \mathbf{C}\ ]$, where $\mathbf{C}$ is appended to $\mathbf{M}$ as the last column (Gauss-Jordan Elimination).

When A, B and C are in standard basis representation, (4) can be written as

$$\begin{bmatrix} a_0^{(0)} & a_0^{(1)} & \cdots & a_0^{(m-1)} \\ a_1^{(0)} & a_1^{(1)} & \cdots & a_1^{(m-1)} \\ & \cdots & \cdots & \\ a_{m-1}^{(0)} & a_{m-1}^{(1)} & \cdots & a_{m-1}^{(m-1)} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \cdots \\ b_{m-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \cdots \\ c_{m-1} \end{bmatrix} \quad (5)$$

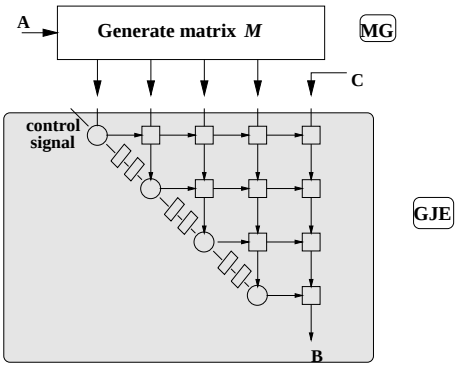

Figure 1. System Diagram of Divider Architecture

where $a_i^{(k)}$ denotes the i-th coordinate of $A\alpha^k \ mod \ f(x)$, b_i denotes the coordinate of B and c_i denotes the coordinate of C, $0 \le i \le m-1$ and $0 \le k \le m-1$. For standard basis representation, all elements in matrix $\mathbf{M}$ need to be computed (except those in the first column which are the coordinates of A).

Dual basis multiplication and division are actually hybrid operations. The multiplicand and the result are in dual basis while the multiplier is in standard basis. Let A and C are expressed in dual basis and B is in standard basis. Let a_i be the coordinates of A in dual basis and $a_i = Tr(\theta A\alpha^i)$. Let b_i be the coordinates of B in standard basis and c_i be the coordinates of C in dual basis. Then (4) can be written as

$$\begin{bmatrix} a_0 & a_1 & \cdots & a_{m-1} \\ a_1 & a_2 & \cdots & a_m \\ & \cdots & \cdots & \\ a_{m-1} & a_m & \cdots & a_{2m-2} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \cdots \\ b_{m-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \cdots \\ c_{m-1} \end{bmatrix}. \quad (6)$$

Due to the properties of dual basis, **MB=C** becomes a discrete-time Wiener-Hopf equation and only $2m-1$ elements in matrix $\mathbf{M}$ need to be computed (including m elements of coordinates of A).

3. DIVIDER ARCHITECTURE

3.1. Existing Standard and Dual Basis Dividers

One standard basis divider has been proposed in [3] and one dual basis divider has been proposed in [4]. They are based on equations (5) and (6), respectively. Although the basis representations are different, the divider architectures in [3] and [4] are quite similar. Their system level diagram (for $GF(2^4)$) is shown in Figure 1, in which **MG** stands for Matrix Generation, and **GJE** stands for Gauss-Jordan Elimination. The architecture for solving the system of equations using Gauss elimination and back substitution (Gauss-Jordan Elimination) is the same for both previous architectures as well as the architectures in this paper. As to the matrix generation part, the two existing dividers both have m basic cells and each basic cell consists of one AND gate, one XOR gate, two MUXes and six latches. In order to synchronize the output from the matrix generation part to satisfy the requirement of the Gauss-Jordan elimination part, $(m^2-m)/2$ latches have been added. Thus the total hardware complexity is m AND gates, m XOR gates,

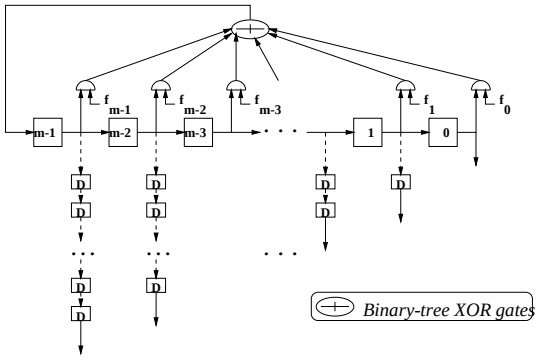

Figure 2. Architecture 1: Generating the Matrix M

$2m$ MUXes and $(m^2+11m)/2$ latches. As can be seen, the $O(m^2)$ latches add a lot of overhead as m increases. In addition, the advantage of dual basis representation where only $m-1$ elements need to be computed to form the system of equations (matrix $\mathbf{M}$) has not been made use of.

Based on above observations, two new dual basis divider architectures are proposed which minimize the hardware complexity by taking advantage of the dual basis representation.

3.2. Proposed Dual Basis Divider Architectures

Architecture 1.

The first proposed architecture for matrix generation part is shown in Figure 2 (in which f_i denotes the coefficient of the primitive polynomial $f(x)$, $0 \le i \le m-1$). The upper part of linear feedback shift register (LFSR) in Figure 2 is used to generate $m-1$ elements of matrix $\mathbf{M}$ and the second part of latches is used to provide appropriate delays such that the output from this section are in the following form which is required by the Gauss-Jordan array:

$$\begin{array}{cccccc} 0 & & & & & c_3 \\ 0 & & & & a_6 & c_2 \\ 0 & & & a_5 & a_5 & c_1 \\ 1 & & a_4 & a_4 & a_4 & c_0 \\ 0 & a_3 & a_3 & a_3 & a_3 & \\ 0 & a_2 & a_2 & a_2 & & \\ 0 & a_1 & a_1 & & & \\ 1 & a_0 & & & & \end{array} \qquad (7)$$

where the first column denotes the control signal. The coordinates of divisor A are loaded to the LFSR in parallel every m clock cycles. The same control signal for Gauss-Jordan array can be used as the select signal for input multiplexing. The hardware requirement is m AND gates, $m-1$ XOR gates, m latches and $(m^2+m)/2$ latches. This architecture is efficient only when $m \le 4$. ■

Architecture 2.

As can be seen from (7), at each time instance the inputs to different columns of Gauss-Jordan array are the same. This implies that broadcast signal can be used to further reduce the number of latches. The second proposed architecture for matrix generation part is shown in Figure 3. The upper part of LFSR remains the same as that in Architecture 1 while the hardware complexity of the second part has

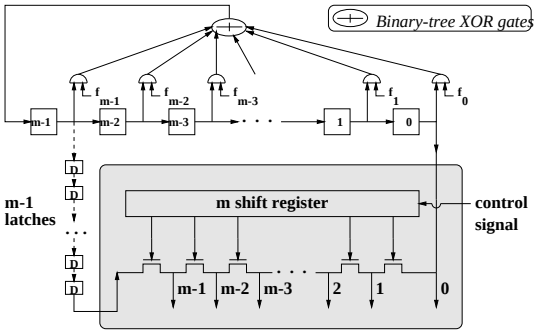

Figure 3. Architecture 2: Generating the Matrix M

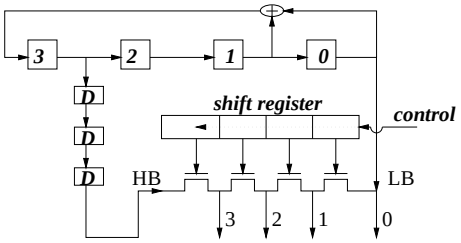

Figure 4. Matrix Generation Part for Divider in Example 1

been reduced from $O(m^2)$ to $O(m)$. In this architecture, the advantage of dual basis representation has been fully utilized and only $2m-1$ latches are required for computing and storing the elements of matrix $\mathbf{M}$. The shaded region in Figure 3 is used to perform conditional broadcasting of the signals from both sides and the broadcasting path is controlled by the control signal in the m shift registers. (The same control signal for input multiplexing can be used in the shaded region.) This portion is necessary because the divider is a pipelined structure and several computations can be performed in a pipeline interleaved way. This architecture is explained in more detail in Example 1. ■

Example 1. This example shows how the second proposed architecture works in a $GF(2^4)$ divider. Let $f(x) = x^4+x+1$ be the primitive polynomial for $GF(2^4)$. Then the architecture for matrix generation is derived and shown in Figure 4. The contents of the m-bit shift register and the output of this structure are listed in Figure 5. Initially, the

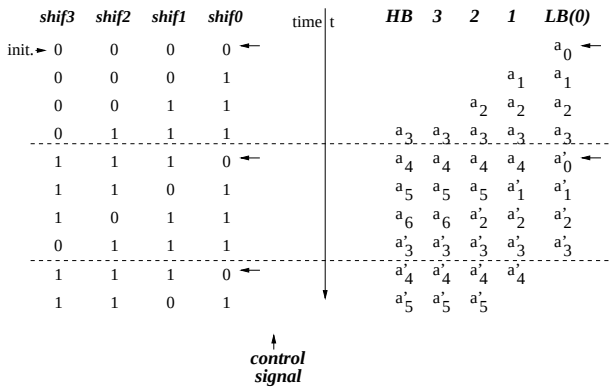

Figure 5. Tabular Form for Example 1

4 registers are reset to zero and all the pass-transistors are off. At this time, only a_0 is output. In the next 3 clock cycles, as the control signals are shifted through the registers, a_1, a_2 and a_3 are broadcast from the right side during the 3 following clock cycles while the path from the left side is off. Then at 5-th clock cycle, a new set of inputs are loaded and another computation is performed concurrently with the previous one. In this clock cycle, the control signal 0 turns off the rightmost pass-transistor and two broadcast signals, one from the left side for the previous computation and one from the right side for the new computation, are broadcast simultaneously through the pass-transistors under the control of the contents of the shift register for another 4 consecutive clock cycles. Thus the computation has been performed in a pipeline interleaved fashion.

The total number of latches has been reduced to $3m-1$ in Architecture 2 while the latency and throughput remain the same as the existing designs. Furthermore, only one control signal is required to load inputs, generate matrix **M** and compute Gauss-Jordan elimination. ■

It needs to be pointed out that for $m \leq 4$, Architecture 1 is more efficient and for $m > 4$, Architecture 2 is more efficient. Since dual basis multiplication and division are hybrid operations using both standard basis and dual basis representation, basis conversion need to be performed very often. It has been proved in [8] and [4] that by appropriately selecting element θ and the corresponding dual basis, basis conversion can be performed by permutation with little or no extra hardware. Hence the basis conversion is no longer a problem as long as the primitive polynomial can be preselected and fixed.

4. SUBSTRUCTURE SHARING

Finite field arithmetic architectures are used as building blocks for large systems, such as Reed-Solomon decoder, which contains the structure that several elements are divided by one element at the same time. Assume that i elements $C1, C2, \cdots, Ci$ are divided by element A at the same time, as shown in Figure 6(a). The division algorithm of solving system of linear equations is proceeded by performing elementary row operations on matrix [**M C**], where the coordinates of dividend C are appended to **M** as the last column. To this end, in order to perform these simultaneous divisions, the coordinates of $C2, C3, \cdots, Ci$ can be input to Gauss-Jordan elimination array (GJE) at the same time as $C1$ (i.e., appended as extra columns to matrix [**M C1**]) and the same elementary row operations will be performed for all of them. Hence by simply adding $i-1$ columns of square cells, $i-1$ more division operations can be performed at the same time, which in turn saves $i-1$ copies of matrix generation part and GJE part as shown in the shaded region in Figure 6.

5. CONCLUSIONS

In this paper, a new dual basis finite field divider has been presented. This new architecture has the same latency and throughput as the existing dividers in [3] and [4]. However, the hardware complexity has been reduced dramatically. A sub-structure sharing technique has also been presented and large amount of savings in hardware can be achieved when this divider is used as a building block for large systems.

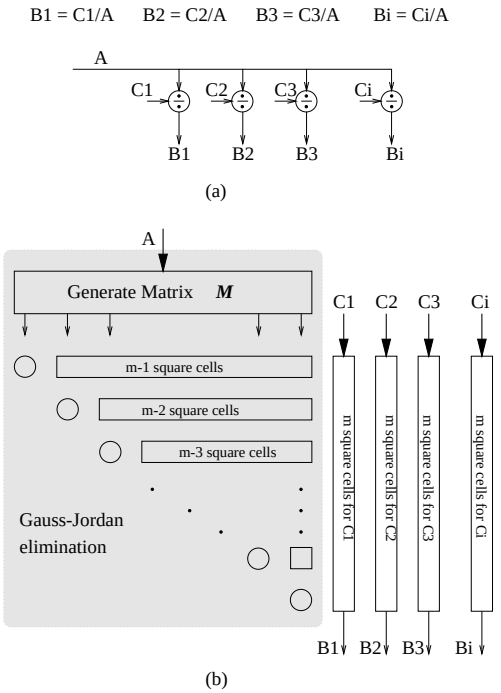

Figure 6. Substructure Sharing for Divider Blocks in a Large System

REFERENCES

[1] C. C. Wang et al, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)", *IEEE Trans. on Computers*, vol. c-34, pp. 709–716, August 1985.

[2] S. T. J. Fenn, M. Benaissa, and D. Taylor, "Finite Field Inversion over the Dual Basis", *IEEE Trans. on VLSI systems*, vol. 4, pp. 134–137, March 1996.

[3] M. A. Hasan and V. K. Bhargava, "Division and bit-serial multiplication over GF(q^m)", *IEE Proceedings-E*, vol. 139, pp. 230–236, May 1992.

[4] S. T.J. Fenn, M. Benaissa, and D. Taylor, "$GF(2^m)$ Multiplication and Division Over the Dual Basis", *IEEE Trans. on Computers*, vol. 45, pp. 319–327, March 1996.

[5] R. E. Blahut, *Theory and Pratice of Error Control Codes*, Addison Wesley, 1984.

[6] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*, The MIT Press, 1972.

[7] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic, 1987.

[8] M. Morii, M. Kasahara, and D. L. Whiting, "Efficient Bit-serial Multiplication and the Discrete-Time Wiener-Hopf Equation Over Finite Field", *IEEE Trans. on Information Theory*, vol. 35, pp. 1177–1183, Nov 1989.