LOW LATENCY STANDARD BASIS GF(2^M) MULTIPLIER AND SQUARER ARCHITECTURES

Surendra K. Jain and Keshab K. Parhi

Department of Electrical Engineering
University of Minnesota, Minneapolis, MN 55455

ABSTRACT

A new parallel-in-parallel-out bit-level pipelined multiplier is presented to perform multiplication in GF(2^m). This new multiplier uses m^2 basic cells where each cell has 2 2-input AND, 2 2-input XOR and 3 1-bit latches. The system latency of this multiplier is $m+1$ compared to $3m$ in previous architectures. The number of latches required per cell has also been reduced from 7 to 3. We also present a bit-level pipelined parallel-in-parallel-out squarer. This squarer has a system latency of $\lfloor m/2 \rfloor$ compared to $3m$ in previous designs and is 25 % more hardware efficient. The critical paths in both these proposed designs are the same as in existing designs.

1. INTRODUCTION

In recent years, finite fields have received a lot of attention because of their application in error control coding [1] [2]. They have also been used in digital signal processing, pseudorandom number generation, encryption and decryption protocols in cryptography. The design of circuits with low circuit complexity, short computation delay and latency to perform finite field arithmetic operations is a matter of great practical concern. Addition in finite fields is bit independent and is a relatively straightforward operation. However, multiplication is a more complex operation. The elements of GF(2^m) can be represented in standard, normal or dual basis [1] [2]. The standard basis multipliers have lower design complexity and are easier to extend to larger finite fields because of their simplicity, modularity and regularity in architecture.

In this paper, we present a new multiplier using standard basis representation. The proposed multiplier has a latency of $m+1$ compared to the existing designs which have a latency of $3m$. The proposed multiplier needs m^2 identical cells, each of which needs two 2-input AND gates, two 2-input XOR gates and three one bit latches. The delay involved in the circuit is the propagation delay of a 2-input AND gate followed by a 2-input XOR gate.

We also present a reduced complexity squarer. The squarer consists of $m\lfloor m/2 \rfloor$ cells, where each cell has 3 2-input AND, 3 2-input XOR and 4 1-bit latches. This squarer results in hardware savings of 25 % and a reduction in latency from $3m$ to $\lfloor m/2 \rfloor$. The critical path in the squarer is the delay of a 2-input AND gate followed by a 3-input XOR gate.

2. ALGORITHM

It is assumed that the reader is familiar with the basic concepts of finite field. The properties of finite fields are covered in detail in [1] [2].

Let $A = \Sigma_{k=0}^{m-1} a_k \alpha^k$ and $B = \Sigma_{k=0}^{m-1} b_k \alpha^k$ be two elements of GF(2^m), $P = \Sigma_{k=0}^{m-1} p_k \alpha^k$ be the product of A and B and $F = \Sigma_{k=0}^{m-1} f_k \alpha^k$ be the irreducible polynomial in standard basis representation. The equation $F = 0$ or $\alpha^m = \Sigma_{k=0}^{m-1} f_k \alpha^k$ is used to reduce the product $P = AB$ to a polynomial of degree less than m. P can be written as follows.

$$\begin{aligned} P &= AB = \Sigma_{k=0}^{m-1} (A\alpha^k) b_k \\ &= (\ldots((Ab_{m-1}\alpha + Ab_{m-2})\alpha + \ldots)\alpha + Ab_0 \end{aligned} \quad (1)$$

Let $A\alpha^k = \Sigma_{n=0}^{m-1} a_n^k \alpha^n$, i.e., a_n^k is the coefficient of α^n when A is multiplied by α^k. The computation of $A\alpha^k$ can be performed recursively on k for $0 \leq k \leq m-1$.

Initially, $A\alpha^0 = A$, i.e., $a_n^0 = a_n$ for $0 \leq n \leq m-1$. For $1 \leq k \leq m-1$, using properties of finite fields, it can be shown [1] [2] that

$$\begin{aligned} a_n^k &= a_{n-1}^{k-1} + a_{m-1}^{k-1} f_n, 1 \leq n \leq m-1 \\ a_0^k &= a_{m-1}^{k-1} f_0. \end{aligned} \quad (2)$$

This can be extended to obtain $A\alpha^{2k}$, for which we get

$$\begin{aligned} a_n^{2k} &= a_{n-2}^{2k-2} + a_{m-1}^{2k-2} f_n' + a_{m-2}^{2k-2} f_n, 2 \leq n \leq m-1 \\ a_0^{2k} &= a_{m-1}^{2k-2} f_0' + a_{m-2}^{2k-2} f_0 \\ a_1^{2k} &= a_{m-1}^{2k-2} f_1' + a_{m-2}^{2k-2} f_1 \end{aligned} \quad (3)$$

This research was supported by Advanced Research Project Agency and the Solid State Electronics Directorate, Wright-Patterson AFB under contract number AF/F33615-93-C-1309.

0-7803-2431-5/95 $4.00

where f' is αf.

3. A PARALLEL-IN-PARALLEL-OUT MULTIPLIER

This multiplication algorithm can be illustrated by a Dependence Graph as shown in Figure 1 where m^2 basic cells are used.

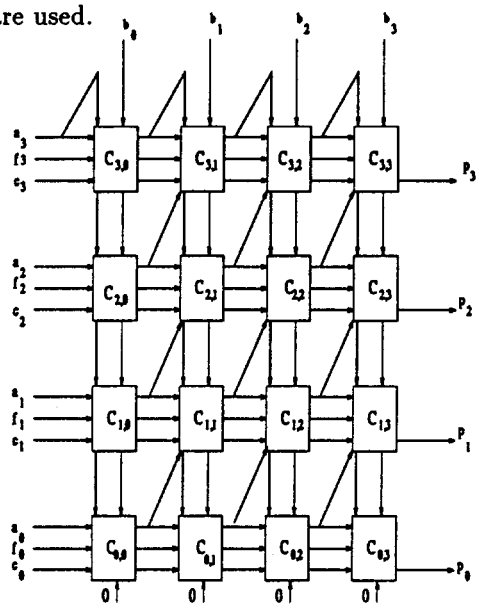

Figure 1: **Dependence Graph for the Multiplication Algorithm**

The basic cell $C_{i,j}$ at position (i,j) receives a_i^j(coefficient of α^i in $A\alpha^j$, f_i (coefficient of α^i in the irreducible polynomial, c_i^j ($= \Sigma_i a_i^{j-1} b_i$), a_{i-1}^j (coefficient of α^{i-1} in $A\alpha^j$, u_j (coefficient of α^{m-1} in $A\alpha^j$) and b_j (coefficient of α^j in B). This cell computes a_i^{j+1} and $c_i^{j+1} = \Sigma_i a_i^j b_i$: This basic cell is shown in Figure 2.

We can now pipeline the Dependence Graph (DG) shown in Figure 1 to obtain a bit-level-pipelined parallel multiplier architecture. We can use the retiming algorithm [3] assuming infinite delays are available at the input for pipelining. The cutsets for a bit-level pipelined architecture where the critical path is the computational delay of 1 basic cell is shown in Figure 3.

The critical path with these cutsets is equal to the delay of a 2-input AND followed by a 2-input XOR gate. Notice that a_i^{j+1} output from cell $C_{i,j}$ feeds both $C_{i,j+1}$ and $C_{i+1,j+1}$, we can therefore make the DFG more regular by removing the diagonal connection and feeding a_i^{j+1} to cell $C_{i+1,j+1}$ from the cell $C_{i,j+1}$. We can also move the pipelining delays inside the basic cell. The modified basic cell is shown in Figure 4.

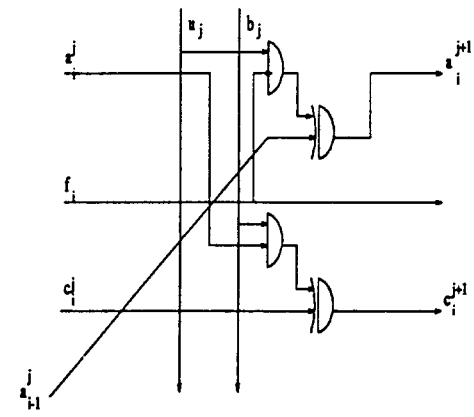

Figure 2: **Basic Cell $C_{i,j}$ of the proposed multiplier**

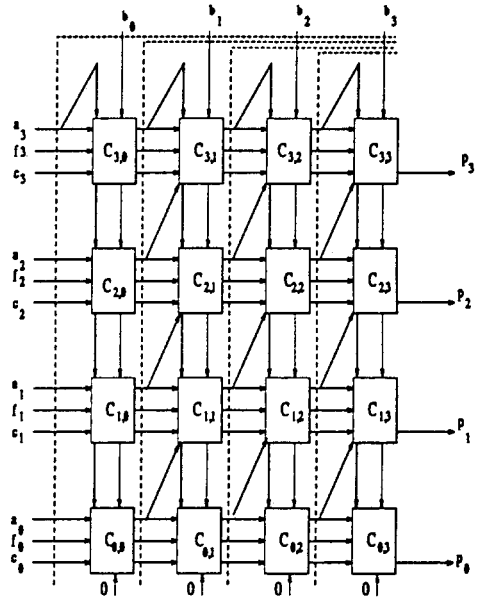

Figure 3: **Cutsets for Pipelining the DG**

Using this basic cell, we can build a parallel-in-parallel-out multiplier which has a latency of $m+1$. A system level diagram of a parallel-in-parallel-out multiplier with this basic cell for GF(2^4) is shown in Figure 5.

The multiplier has m^2 cells and each cell has 2 2-input AND, 2 2-input XOR gates and 3 1-bit latches. The parameter $C = \Sigma_{k=0}^{m-1} c_k \alpha^k$, an element of GF($2^m$), is also input to the multiplier so that the circuit actually performs $AB + C$.

Notice that in the parallel multiplier $v_{m,j}$ is connected to u_j. It is worth noting that although it appears in the DFG that edges in the vertical direction are going in both up and down directions, that is not the case. The edge from cell $C_{i,j+1}$ to the cell $C_{i+1,j+1}$ is actually the output of the cell $C_{i,j}$ and hence there

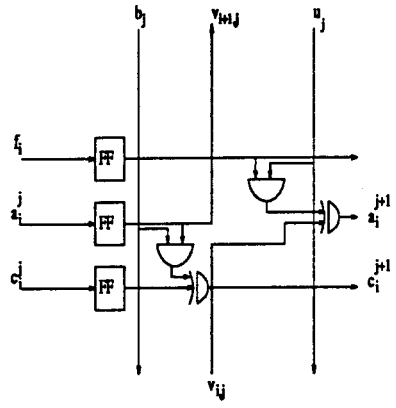

Figure 4: **Modified Basic Cell $C_{i,j}$ of the proposed multiplier**

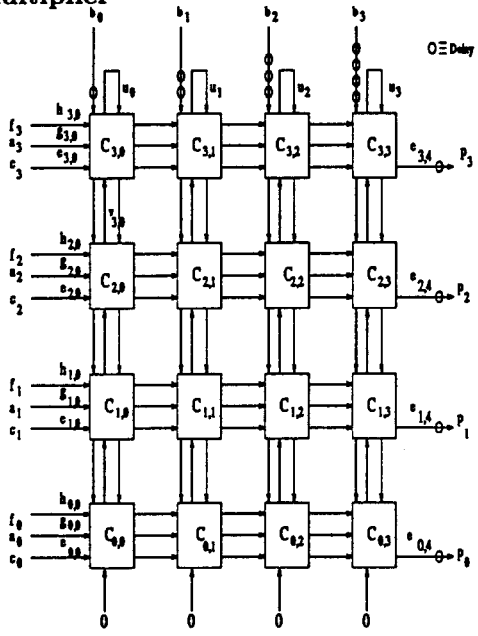

Figure 5: **Parallel-in-Parallel-out Multiplier**

is no feedback path in this DFG. This modification is made to make the DFG more regular and hence more suitable for VLSI implementation.

Inputs $e'_{n,0}s, g'_{n,0}s$ and $h'_{n,0}s$ receive in parallel the $c'_n s$ of C, $a'_n s$ of A, $f'_n s$ of F, respectively, for $0 \leq n \leq 3$. Th e $p'_n s$ of the result P are transmitted out in parallel from the outputs $e'_{n,4}s$ for $0 \leq n \leq 3$. One may use degenerate versions of the cell shown in Figure 4 in the bottom row and the rightmost column of the multiplier since some of the inputs and outputs of these cells are not used.

The proposed multiplier is a semi-systolic version of the multiplier of [4]. The signals b and u in this design have been made broadcast signals. The design of the multiplier of [4] can also be obtained from the Dependence Graph (DG) shown in Figure 1. We can again assume infinite number of delays at the input and obtain a different retiming solution to obtain the systolic version proposed in [4].

4. A PARALLEL-IN-PARALLEL-OUT SQUARER

In a finite field,

$$(\alpha + \beta)^2 = \alpha^2 + \beta^2 \tag{4}$$

where $\alpha, \beta \epsilon GF$. Using this property of finite field and equation (3), we can develop a hardware efficient squarer for finite field. We shall illustrate this with an example for GF(2^4). Squaring operation can be represented by

$$A = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 \tag{5}$$
$$A^2 = a_0 + a_1\alpha^2 + a_2\alpha^4 + a_3\alpha^6. \tag{6}$$

To obtain the result in the standard basis, we need

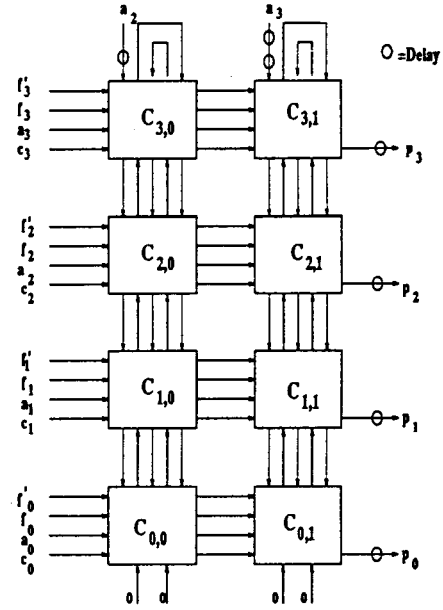

Figure 6: **Parallel-in-Parallel-out Squarer**

to express α^4, α^6 in terms of $1, \alpha, \alpha^2, \alpha^3$. This can be achieved using the squarer shown in Figure 6. The squarer has been designed using the technique illustrated in Section 3 for the multiplier. The squarer consists of $m\lfloor m/2 \rfloor$ basic cells. The inputs to the squarer are $C = a_0 + a_1\alpha^2$, $B = \alpha^4$, a_2, a_3, f and f'. The first column computes $Ba_2 + C$ and $B\alpha^2$ while the second column outputs the desired result A^2.

The basic cell performs the step described in equation (3) and is shown in Figure 7. The squarer is semi-systolic where each basic cell needs 4 latches. A fully systolic version would need 10 latches [5]. The critical

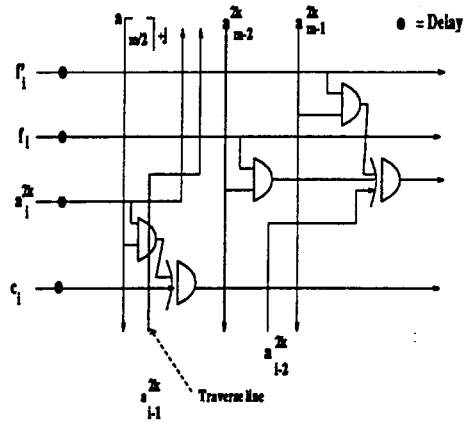

Figure 7: **Basic Cell $C_{i,j}$ of the squarer**

path in both the cases is equal to the delay of 2 input AND and 3 input XOR gate. This squarer design is easily extendable to a larger finite field. In general, for GF(2^m), we need $\lfloor m/2 \rfloor$ columns where each column comprises of m basic cells. Note that we can use degenerate versions of the basic cell in the rightmost column and in the bottom row because some of the outputs are not needed. Again, there are no feedback paths in the squarer.

5. CONCLUSIONS

The properties of the proposed multiplier are compared in Table 1 with those of the multipliers of [4] [6].

Item	Yeh *et. al*	Wang *et. al*	Figure 5
Number of basic cells	m^2	m^2	m^2
Basic Cell	2 2-input AND, 2 2-input XOR, 7 1-bit latches	2 2-input AND, 1 3-input XOR 7 1-bit latches	2 2-input AND, 2 2-inputXOR 3 1-bit latches
Latency	3m	3m	m+1
Time step	1 2-input AND and 1 2-input XOR gate	1 2-input AND and 1 3-input XOR gate	1 2-input AND and 1 2-input XOR gate

Table 1: **Comparison of Different Multipliers**

Table 2 compares the proposed squarer with using a dedicated multiplier and the power-sum circuit [5]. It is worth noting that the proposed multiplier needs less than half the number of latches required in previous implementations while maintaining the same critical path. The system latency has also been reduced to $m+1$ from $3m$. The price we pay for this reduction in hardware requirement and system latency is to allow two broadcast signals.

The proposed squarer results in hardware savings of more than 50 % over using the power-sum circuit of [5]

Item	Multiplier	Power-sum	Figure 4
Number of basic cells	m^2	m^2	$m\lfloor m/2 \rfloor$
Basic Cell	2 2-input AND, 2 2-input XOR, 3 1-bit latches	3 2-input AND, 3 2-input XOR 10 1-bit latches	3 2-input AND, 3 2-inputXOR 4 1-bit latches
Latency	3m	3m	$\lfloor m/2 \rfloor$
Time step	1 2-input AND and 1 2-input XOR gate	1 2-input AND and 1 3-input XOR gate	1 2-input AND and 1 3-input XOR gate

Table 2: **Comparison of Different Approaches to Squaring**

and savings of more than 25 % over a dedicated multiplier to perform the squaring operation. The system latency has been reduced to $\lfloor m/2 \rfloor$ from $3m$ without any increase in the critical path.

The proposed multiplier and squarer, if used, in the square and multiply algorithm to perform exponentiation would result in hardware savings of over 12.5 % over the present design [7] and reduction in system latency to m^2+1 from $2m^2+m$. Both the proposed multiplier and squarer are well suited for VLSI systems because of regular interconnection pattern, modular structure and complete concurrency in operations.

6. REFERENCES

[1] R. Blahut, *Theory and Practice of Error Control Codes.* Addison-Wesley Publishing Company, 1983.

[2] F. J. MacWilliams and N. J. A. Sloane, *Theory of Error Correcting Codes.* New-York:North-Holland, 1977.

[3] C. E. Leiserson, F. Rose, and J. Saxe, "Optimizing synchronous circuits by retiming," *Proc. of the third Caltech Conference on VLSI*, pp. 87–116, March 1983.

[4] C. S. Yeh, I. S. Reed, and T. K. Truong, "Systolic multipliers for finite fields GF(2^m)," *IEEE Transactions on Computers*, vol. C-33, no. 4, pp. 357–360, April 1984.

[5] S.-W. Wei, "A systolic power-sum circuit for GF(2^m)," *IEEE Transactions on Computers*, vol. 43, no. 2, pp. 226–229, Feb. 1994.

[6] C. L. Wang and J. L. Lin, "Systolic array implementation of multipliers for finite fields GF(2^m)," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 7, pp. 796–800, July 1991.

[7] C.-L. Wang, "Bit-level systolic array for fast exponentiation in GF(2^m)," *IEEE Transactions on Computers*, vol. 43, no. 7, pp. 838–841, July 1994.