Low-Power Design of Finite Field Multipliers for Wireless Applications*

A. G. Wassal, M. A. Hassan and M. I. Elmasry
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON N2L 3G1

Abstract

Unlike most research involving finite field multipliers, this work targets a low-power multiplier through the application of various power reduction techniques to different types of multipliers and comparing their power consumption among other factors, rather than comparing complexity measures such as gate count or area. Gate count is used as a starting point to choose potential architectures, namely, polynomial and normal basis architectures. Power reduction techniques employed are mainly concerned with Architecture- and Logic-Level low-power techniques. They include supply voltage reduction, power cost estimations, using low-power logic families and pipelining.

1 Introduction

Recently, finite field arithmetic has found wide use in different applications such as error-control codes, cryptography, switching theory, and digital signal processing. However, its use in channel coding and cryptography modules for wireless communications has been the main motive for a low-power implementation. Wireless devices are based on a rechargeable battery supply. Obviously, the less the power dissipated, the longer the battery life and the more operation time the user gets from the wireless device before a battery recharge. Multiplication is at the heart of almost all finite field operations. Consequently, a low-power finite field multiplier is essential.

A mathematical background of the finite fields is necessary to understand the design of the multiplier and how it works [1]. A few definitions are in order here. An *abelian group* G is a set of elements together with a binary operation $*$ satisfying the following mathematical properties: closure, associativity, having an identity element, having inverses and commutativity. A *field* is a set F together with two operations, addition and multiplication such that F is an abelian group under addition with 0 as the identity element, the non-zero elements of F form an abelian group under multiplication with 1 as the identity element and the distributive law $a(b+c) = ab + ac$ holds for all a, b, c in the field. A field with a finite number of elements p is called a *finite field* of order p and is usually denoted by GF(p), i.e., Galois Field, or finite field, of order p. The order, p, must be a prime number to ensure that the field is a group under modulo-p multiplication. The binary field GF(2) and its extension GF(2^m) are the most used fields in computers and communications.

Finite field arithmetic operations are essentially implemented in all architectures using XOR gates and registers. In fact, addition in GF(2) is nothing but an XOR operation. It is also worth noting that there is no carry to propagate in finite field arithmetic operations which means a smaller critical path compared to ordinary arithmetic operations. A building block for almost all finite field operations is a finite field multiplier.

There are two parameters which are of central importance for computer arithmetic finite field multiplier implementations, the speed which is specified by the maximum clock rate or the maximum delay, and the complexity. The vast majority of the literature dealing with finite field multipliers uses the critical path as a measure of the delay and the gate count as a measure of the area and complexity and tries to minimize them [2, 3]. However, this work tries to minimize power dissipation in the multiplier since we target a Low-Power implementation suitable for wireless applications. At the same time, we will also consider acceptable ranges for the other parameters, namely, delay and gate count. The maximum delay is measured and optimized for single primitive gates and the critical path is used to optimize the whole design using automatic synthesis optimization tools. Also, the

*This work has been supported in part by MICRONET.

0-8186-8409-7/98 $10.00 © 1998 IEEE

gate count is used among other properties to select the initial architectures and their initial design styles, and then these are optimized with respect to power consumption.

In section 2, the gate count is used as a measure of complexity to find a starting point for choosing architectures with potential for power reduction, namely, polynomial and normal basis architectures. This section also discusses signal distribution and pipelining issues. Power reduction techniques that can be applied to finite field multipliers are discussed in section 3 and some of them are employed to reduce the power consumption of the chosen architectures. The techniques used include supply voltage reduction, power cost estimations and use of low-power logic families. Section 4 reviews the simulations and characterization of the designs. Finally, results are summarized and conclusions are drawn in section 5.

2 Low-Power Architecture Design

The architectures of the finite field multipliers are usually classified according to their representation of the field elements. There are three popular representations which have been utilized, namely the polynomial (canonical or standard), the normal and the dual basis representation. The different bases yield quite different architectures [2].

2.1 Architecture Selection

Table 1 shows some of the key parameters for different architectures of the finite field multipliers over the extension field $GF(2^m)$ [3, 2, 4, 5, 6]. It can be noted that the normal basis requires the least number of gates, unfortunately this is for *optimal* normal basis multipliers which can be realized for only $\approx 23\%$ of the fields GF(2^m), $2 \leq m < 1200$ [7]. Also, the dual basis multiplier requires conversion of one of its inputs to a representation different from the other input which requires extra gates. Another point to consider is that both the normal and dual bases are not highly modular or expandable. Based on that comparison, the polynomial basis architecture seems to be a good initial architecture [6]. This architecture is highly modular and expandable so it can be implemented as a 1-bit multiplier slice and then any field order can be realized. Also, if we assume that we will use only those orders that can be implemented using an *optimal* normal basis architecture and if we restrict our comparison to those specific field orders, then we can use the normal basis architecture too. The polynomial basis multiplier can be implemented using different architectures in its turn [2], the most

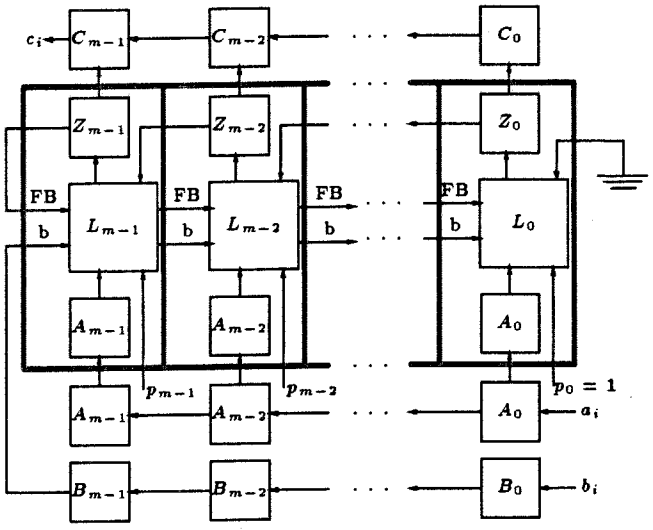

Figure 1: The SSR polynomial architecture with the slices highlighted

common implementations are the Standard Shift Register (SSR) and the Modified Shift Register (MSR) shown in figures 1 and 3. Both implementations can be divided into bit-slices and we can then replicate as many of these bit-slices as we want and provide the irreducible polynomial coefficients P_i to produce the required multiplier. Each bit-slice, as highlighted in figures 1 and 3, is basically composed of an input register, whether it is a normal register or an LFSR, an output register that acts as a GF(2) accumulator and a combinational logic block in between them. The combinational logic block, figures 2 and 4, will be called the *Lcell*. Since the gate count is almost the same, the best implementation is the one that consumes less power. Also, the MSR implementation has a slightly shorter critical path in terms of gate count than the SSR implementation. The normal basis architecture shown in figure 5 is not as regular and modular as the polynomial one and it does not render itself to bit-slice implementations. Hence, we will use normal multipliers implemented over GF(2^4) [5] and 4 bit-

Basis	Polynomial	Normal	Dual
Complexity Gates/Registers	$4m/2m$	$\geq 3m - 1/2m$	$4m - 2/2m$
Length of CP	4	$\geq 3 + \lceil \log_2 m \rceil$	$\geq 2 + \lceil \log_2 m \rceil$
		$\geq \lceil \log_2 m \rceil$	
Regularity	High	Low	Low
Expandability	High	Low	Low

Table 1: The main properties of different architectures of $GF(2^m)$ multipliers

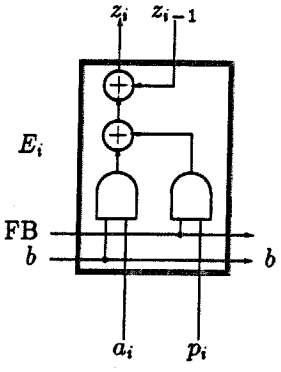

Figure 2: The Lcell structure for the SSR polynomial basis multiplier

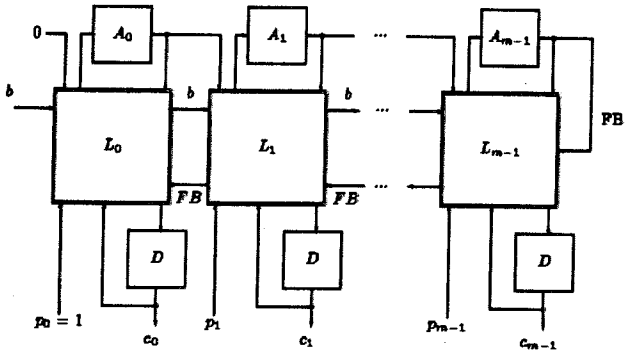

Figure 3: The MSR polynomial architecture with the slices highlighted

slices of the polynomial multipliers to have a fair comparison. GF(2^4) is hardly a useful field for encryption purposes, however, it is very indicative with respect to power consumption. Another reason for choosing that field order will be discussed in section 2.2. Function f in the normal multiplier is again not regular and depends on the field order and the polynomial governing it. Figure 6 shows the logic design of the f function of a normal basis multiplier over the field GF(2^4) and using the trinomial $p(x) = 1 + x^3 + x^4$. It is important to notice that the polynomial architecture will require

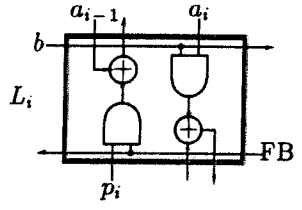

Figure 4: The Lcell structure for the MSR polynomial basis multiplier

fewer gates and hence dissipate less power if it uses a trinomial and if the gates used in the Lcell to provide p_i are eliminated.

We will compare the power costs of the Lcells and the f function based on the switching activity concept as explained in section 3.1. The results shown there endorse the use of the MSR.

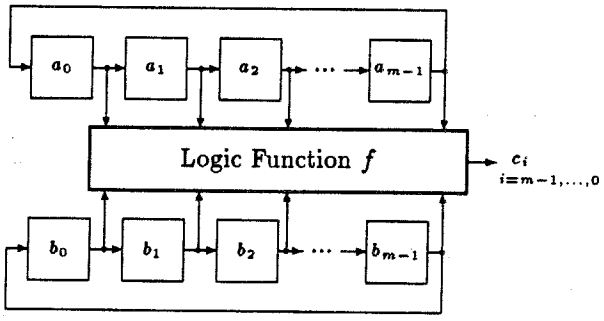

Figure 5: The normal basis multiplier architecture

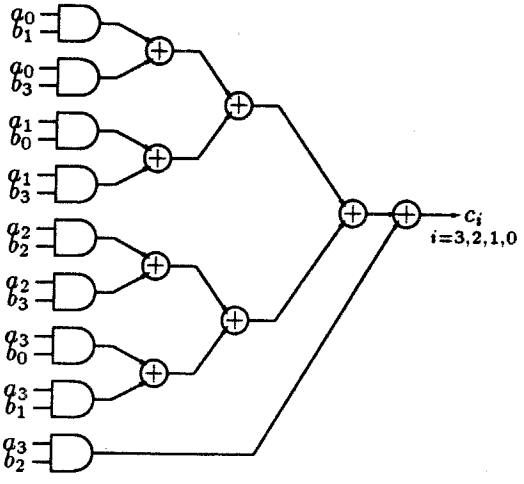

Figure 6: The f function structure for the normal basis multiplier over GF(2^m) with $p(x) = 1 + x^3 + x^4$

2.2 Signal Distribution and Skewing

Signals that are distributed to all slices, such as the clock, FB and b signals, are susceptible to degradation due to their large capacitive loads. The long distribution interconnects of the signal contribute to this load, also the load of the large node capacitances is usually larger than tolerable limits. Most control signals suffer from this problem, but the main obvious one is the clock signal.

To solve this problem, multi-bit slices are proposed for the polynomial basis architecture and buffers are inserted between every slice and the next so as to elevate the problem and reduce the load on the sources of these signals [8]. Figure 7 shows a cost function computed as the prodcut of the delay and power dissipation in the slice versus the number of bits per slice. As shown, the cost function has a minimum at about 4 bits/slice and hence buffers are inserted every four bits on the clock line and on the other control signal lines. Since the normal basis architecture is not implemented in bit-slices, it has this problem for the input registers only. Having chosen 4 bit-slices for the polynomial basis architectures, we will also use a normal basis multiplier implemented over GF(2^4) to be able to have a fair comparison.

A new problem occurs due to the insertion of these buffers and that is the problem of skewed signals. Signals arriving at slices far from their sources pass through a larger number of buffering stages and hence exhibit more delay than the signals arriving at nearer slices. As a result, these signals arriving at different gates or registers are not synchronized and are delayed from each other. To solve this problem, these signals should be distributed using a tree hierarchy of buffers so that the delay would be equal on all paths of the signal and the skew is eliminated. Also, we should try to make the interconnect lengthes in the physical layout as equal as possible, e.g., by making the tree root physically located at the center of the chip and the cells in symmetric locations around the center.

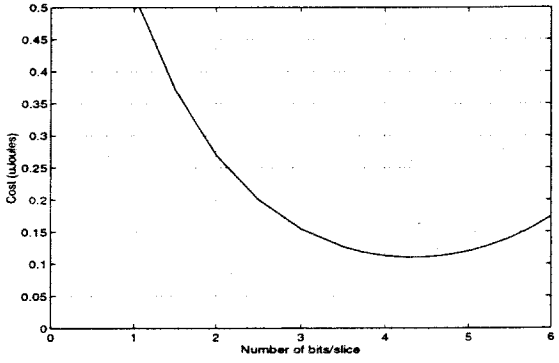

Figure 7: Selecting the number of bits/slice

2.3 Pipelining within the Slice

The original design of the MSR polynomial multiplier used the same clock for the linear feedback shift register (LFSR) and the accumulator which required 2 clock cycles to load a bit and calculate a partial-product. This is not a suitable solution since it halves the throughput of the multiplier. To overcome this problem, we require that the accumulator adds the partial-product fast enough for the LFSR to compute the next partial product within the same clock pulse but this still requires one more clock pulse for the final accumulation. To eliminate this extra cycle, it is better to trigger the LFSR and the accumulator to on different clock edges, that is pipelining within the clock cycle [9]. In our case, the LFSR and the Lcell work within the clock pulse and the accumulator is triggered by the inverted clock and works after the clock pulse, as shown in figure 8. As for the normal basis multiplier, it will be only limited by the critical path through the f function and the registers. This critical path is significant for large finite fields because it will pass through $\log_2 m$ gates of the XOR tree within the f function.

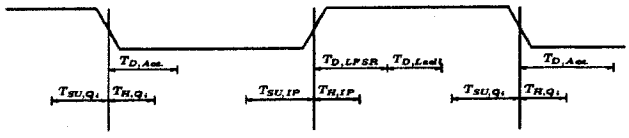

Figure 8: The timing requirements of the polynomial basis clock

3 Logic Design for Low-Power

There are three main power dissipation components in digital designs. These components are dynamic power consumed in the charging and discharging of the circuit capacitances and is the dominant one in CMOS design styles, short circuit power consumed by the direct current from supply to ground during the switching transient and static power caused by leakage current and other static currents. Low-power architecture techniques were discussed in the previous section and this one will discuss gate-level techniques. These are mainly choosing the design that exhibits lower power cost and choosing the design style that achieves the lowest power consumption.

3.1 Power Cost of Different Logic Designs

In this discussion, it is assumed that the main component that contributes to the dissipation of power for the design styles used is the dynamic power. Later, simulations will take other power components into account. For the polynomial architectures, the Lcell is the main part where reduction of power can be achieved since the flip-flops are used from a standard

library and are already optimized for low-power operation. The dynamic power, P_d, dissipated in a basic inverter is expressed as

$$P_d = C_{\text{eff}} V_{\text{dd}}^2 f_{\text{clk}}$$

where C_{eff} is the total effective capacitance of all nodes, V_{dd} is the supply voltage and f_{clk} is the clock frequency. From the above expression, it is obvious that reducing the supply voltage reduces the dynamic power dramatically. The targeted supply voltage for our multiplier is 2.5 V. Reducing the supply voltage below that would require special technology considerations.

To choose the proper implementation of the Lcell, the switching activity [8] is computed for both designs, SSR and MSR, to compare the dynamic power cost for each design style. It is also computed for the f function of the normal basis multiplier. The power cost is equivalent to the total effective capacitance of all nodes, C_{eff}, where the effective capacitance of a node is the actual total capacitance at that node weighted by the probability that this node switches, thus affecting the dynamic power dissipation. The power cost function for the static CMOS is the summation over all nodes i as follows,

$$\text{Power cost} = \sum_i P_{0\to 1,i}\, C_i$$

where $P_{0\to 1,i}$ is the probability that node i switches from logic 0 to logic 1 and C_i is the capacitance of node i. On the other hand, for the Domino logic, the power cost function is as follows,

$$\text{Power cost} = \sum_i P_{0,i}\, C_i$$

where $P_{0,i}$ is the probability that node i evaluates to logic 0 and C_i is the capacitance of node i.

3.2 Selection of the Design Style

The basic gates are implemented as static CMOS, a mixture of static CMOS and transmission gates (TGs), and Domino logic gates. Domino logic and dynamic logic styles in general provide smaller area than fully static gates, smaller parasitic capacitances and glitch free operation if designed carefully [9]. The Domino logic implementation will be chosen if these advantages outweigh the amount of power consumed during the precharge phase. The mixed style is basically static CMOS AND gates and TG based XORs. Using the TGs to implement the AND gates consumes a larger area than the static CMOS case and is not justifiable. However, a TG XOR uses only six transistors

	static CMOS	CMOS+TG	Domino logic
MSR Lcell	89.625 fF	74.625 fF	198.5 fF
SSR Lcell	95.062 fF	87.125 fF	227.5 fF

Table 2: Dynamic power costs for the Lcells

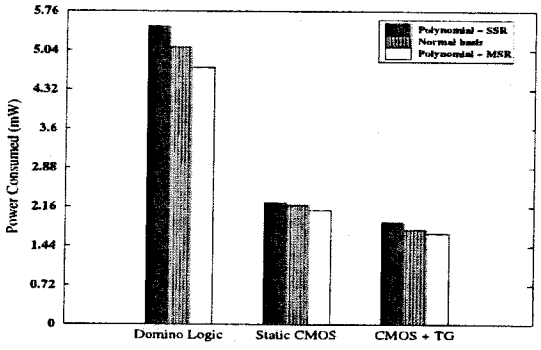

Figure 9: Comparison of power dissipation in different implementations

which is a better implementation than static CMOS implementations in terms of power and area. Afterwards, the capacitances are extracted from the corresponding layouts. Since the inputs can take any value and the registers produce all possible outputs over a sufficient period of time, the inputs to the GF(2^4) Lcells and f function are assumed to be equally probable. Table 2 shows these power costs and it is obvious that even without considering any other type of power dissipation except the dynamic one, the Domino logic design is outperformed by the other logic styles and the MSR design is superior to the SSR design.

The f function could be implemented using a PLA to reduce the design area although that will increase the power dissipation in the standby mode [8].

To select the proper design style for the multiplier, different types of power dissipation components are estimated for the different designs using simulations. The simulations are performed for a 0.8μm technology, at the targeted supply, V_{dd}=2.5v, a nominal load of 2 units, i.e., 0.05 pF and a test frequency of 100MHz. The bar graph in figure 9 shows the power in all of these cases. Again, the obvious candidate is the MSR based polynomial multiplier implemented using the mixture of TGs and static CMOS [8]. Next to the MSR polynomial multiplier comes the normal basis multiplier implemented using the same design style.

4 Simulation and Characterization of the Design

A gate-level simulation uses delay models to verify the correctness of the functionality of the circuit

and its implementation. It also sets a starting point for the frequency of the clock, 10ns clock period i.e., 100MHz. Later, simulation is used to determine the maximum clock frequency that can be used. Also, a fifth clock cycle is used in this simulation to reset the accumulator with every multiplication. This would not be necessary when the multiplier is integrated in an application since there will be an external register to collect the result.

Next, the multipliers are simulated using parametric-simulation and different curves are compiled to characterize their design space with respect to other parameters such as delay and power-delay product. As shown in figure 8, the pulse width of the clock for the polynomial basis multiplier is specified by the delay time of the LFSR plus the delay time of the Lcell and the setup time of the flip-flops, assuming that the flip-flops of the LFSR and the accumulator have the same timing characteristics and that the hold time of the flip-flops, approximately 0.42 nsec, is smaller than the sum of the delay times of the LFSR and the Lcell. Similarly, the delay time of the accumulator and the setup time of the LFSR specify the remainder of the clock period. From figure 10 and at the targeted supply voltage of 2.5v, the sum of LFSR and Lcell delays is approximately 3.6 nsec and the delay of the accumulator is approximately 1.6 nsec while the setup time of the flip-flops is 0.75 nsec. Therefore, the minimum clock period is 6.7 nsec which corresponds to a maximum frequency of approximately 150 MHz. This shows that the clock period used in the gate-level simulations, which was 10 nsec, was a valid one. A similar analysis is used to derive the clock period for the normal basis multiplier. Table 3 includes the results of that analysis.

The compiled curves in figures 10-13 are very useful in determining the proper operating point for different applications. They help the designer to choose the clock frequency, trigger the different parts of the multiplier at the appropriate clock edges, choose the proper supply voltage and have estimates of the power dissipated and the power-delay product in the design.

5 Summary

This work has shown that with careful application of low-power design techniques, finite field multiplier designs customized for the power requirements of wireless applications can be attained. Among the designs discussed, the MSR polynomial basis multiplier implemented in static CMOS and transmission gates dissipated the least amount of power. It is also a highly

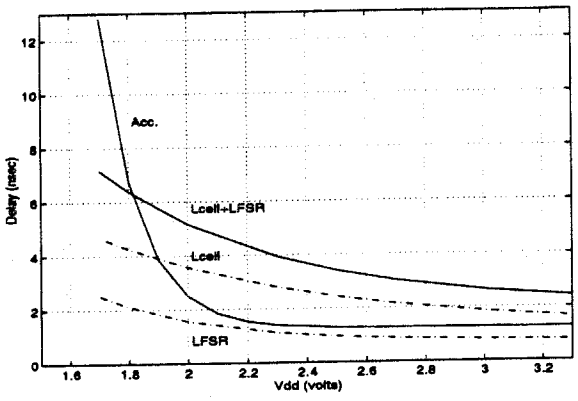

Figure 10: Accumulator, LFSR and Lcell delays vs. V_{dd} @C_L = 0.05pF for the MSR polynomial architecture

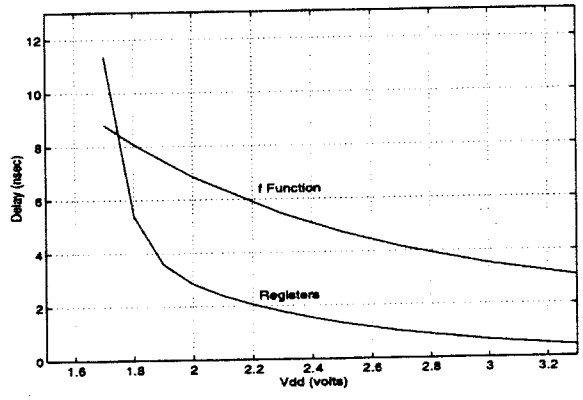

Figure 11: Registers and f function delays vs. V_{dd} @C_L = 0.05pF for the normal basis architecture

modular and regular design that can be customized to the number of bits required by the application within a smaller chip area. The MSR polynomial basis multiplier is also the fastest. The normal basis multiplier implemented using the same logic style comes next. It has a little higher amount of power dissipation and delay. Its main disadvantages are its low modularity and regularity which means larger chip area and more design time and effort. A set of characterization curves has also been used to help defining the design space. Table 3 summarizes the specifications and characterizations of those two designs.

References

[1] Shu Lin and Jr. Daniel J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., 1983.

Feature	MSR Polynomial Basis Multiplier	Normal Basis Multiplier
Supply voltage	2.5 V	2.5 V
Minimum clock period	6.7 nsec	8.1 nsec
Maximum clock frequency	149 MHz	123 MHz
Maximum throughput	37.25 $\times 10^6$ op/sec	30.75 $\times 10^6$ op/sec
Min. clock period (50% duty)	8.7 nsec	10.3 nsec
Max. clock frequency (50% duty)	115 MHz	97 MHz
Max. throughput (50% duty)	28.75 $\times 10^6$ ops/sec	24.27 $\times 10^6$ ops/sec
Power dissipation (including flip-flops)	$\approx$ 3.2 mW	$\approx$ 3.7 mW
Power saving compared to SSR arch.	$\approx$ 5.4%	$\approx$ 3.7%
Power saving compared to pure static CMOS style	$\approx$ 9.1%	$\approx$ 6.4%
Power saving compared to Domino style	$\approx$ 52.2%	$\approx$ 36.7%
Chip area	289μm $\times$ 310μm	322μm $\times$ 348μm
Regularity and modularity	High	Low

Table 3: Specifications of the Low-Power Finite Field Multipliers over GF(2^4) in CMOS+TG implementation

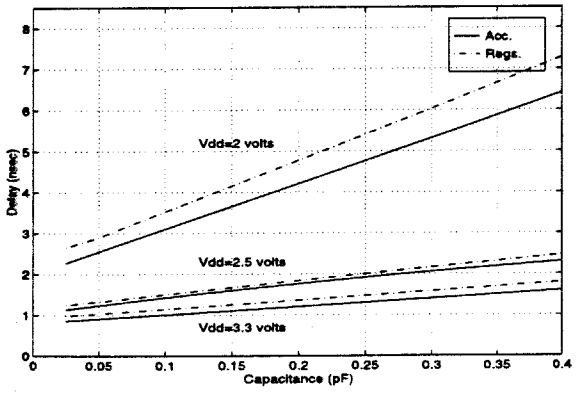

Figure 12: Delay of the accumulator in the MSR polynomial architecture and the registers in the normal architecture vs. Capacitive load (Unit load is approx. 0.025pF)

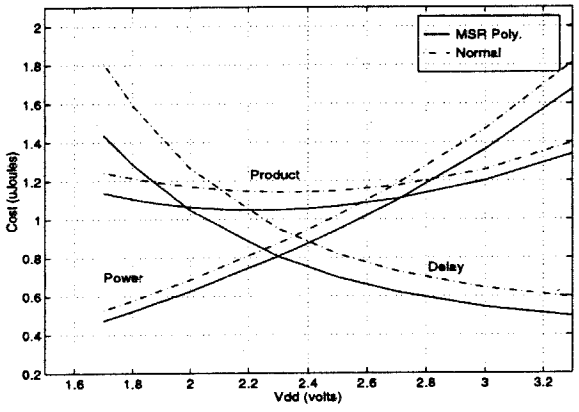

Figure 13: Power-delay product vs. V_{dd} @C_L = 0.05pF (Power and delay curves only show the trend. They are scaled and shifted)

[2] Edoardo D. Mastervito, *VLSI Architectures for Computations in Galois Fields*, Ph.D. thesis, Department of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, 1991.

[3] I. S. Hsu, T. K. Truong, L. J. Deutsch, and I. S. Reed, "A comparison of VLSI architecture of finite field multipliers using dual, normal, or standard bases," *IEEE Transactions on Computers*, vol. 37, no. 6, pp. 735–739, June 1988.

[4] E. Berlekamp, "Bit-serial Reed-Solomon encoders," *IEEE Transactions on Information Theory*, vol. 28, pp. 869–874, November 1982.

[5] J. Massey and J. Omura, "Apparatus for finite field computation," *US Patent Application*, pp. 21–40, 1984.

[6] P. A. Scott, S. E. Tavares, and L. E. Peppard, "A fast multiplier for GF(2^m)," *IEEE Journal on Selected Areas of Communications*, vol. 4, January 1986.

[7] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson, "Optimal normal bases in GF(p^n)," *Discrete Applied Mathematics*, pp. 149–161, 1988/1989.

[8] Abdellatif Bellaouar and Mohamed I. Elmasry, *Low-Power Digital VLSI Design, Circuits and Systems*, Kluwer Academic Publishers, 1995.

[9] Neil H. E. Weste and Kamran Eshraghian, *Principles of CMOS VLSI Design, A systems Perspective*, Addison Wesley, 1994.