Finite Field Multipliers for Ultra-Constrained Environments*

Jorge Guajardo[1], Tim Kerins[2**], Sandeep S. Kumar[1], and Pim Tuyls[1]

[1] Philips Research Laboratories, Eindhoven, The Netherlands
{Jorge.Guajardo,Sandeep.Kumar,Pim.Tuyls}@philips.com
[2] Dept. Electrical and Electronic Engineering, University College Cork, Ireland
timk@rennes.ucc.ie

Abstract. This work introduces a new finite field multiplier based on a modification of the standard Most Significant Digit Element (MSDE) first multiplier introduced in [25]. The modification results in a slight improvement in the area complexity of the multiplier when compared to the work in [5] and in a significant reduction in the critical path of the multiplier. We provide estimates for the area and time complexities of the multiplier. We conclude that the presented design is well suited for implementations of ECC that target ultra-constrained environments such as RFID tags and sensor nodes.

Key Words: finite fields, elliptic curve cryptography, small area implementations, RFID, sensor nodes

1 Introduction

In recent years, we have seen the widespread development of technologies which in one way or another will enable the next revolution after the Internet: the Internet of Things. Examples of such technologies are Radio Frequency Identification (RFID) systems and (ad-hoc) sensor networks. RFID technology is an enabler for applications such as [16]: goods tracking in supply chain management, automated inventory management, automated quality control, access control, and payment systems, but also applications deriving from the interaction of tagged objects with intelligent devices in the home and our surroundings (intelligent refrigerators, washing machines, intelligent posters, *etc.*). Sensor networks have wide range of applications [9] such as surveillance, distributed disaster management systems, emergency response, habitat and environmental

* Presented at the 2nd Benelux Workshop on Information and System Security — WISSEC 2007, September 20-21, 2007, Luxembourg city, Luxembourg.
** Work done while the author was at Philips Research.

monitoring, and monitoring of interactions in wildlife, health-care, and manufacturing process flow.

Such applications give rise to a series of security problems which could be roughly divided into two groups: entity authentication (is node A forwarding the right information? is this an authentic tag?) and privacy (is reader X allowed to access the data stored in tag B? Is an external node allowed to listen in on communication in a private network?). Such problems have forced researchers to develop numerous solutions as Juels [16] describes in the context of RFID systems and Chan and Perrig [8] in the context of sensor networks. In the beginning, many of the proposed solutions were based on new protocols and primitives. In particular, traditional cryptographic primitives were not considered feasible because of their area cost and power consumption. However, recently there has been renewed interest in investigating the feasibility of public-key cryptography and, in particular, elliptic curve cryptography (ECC) engines for such constrained environments as RFID [3,4] and sensor nodes [5].

It is well known that any elliptic curve engine is composed of memory, control logic and an arithmetic logic unit (ALU), whose largest component is a finite field multiplier. In fact, there is a large body of research on finite field multipliers already. For example, characteristic two field architectures have been considered extensively, probably due to the straight forward manner in which elements of $\mathbb{F}_2$ can be represented, i.e., they can be represented by the logical values "0" and "1". In recent years, $\mathbb{F}_{p^m}$ fields, where p is odd, have also gained interest in the research community. Mihălescu [18] and independently Bailey and Paar [1] introduced the concept of Optimal Extension Fields (OEFs) in the context of elliptic curve cryptography. OEFs are fields $\mathbb{F}_{p^m}$ where p is odd and both p and m are chosen to match the particular hardware used to perform the arithmetic, thus allowing for efficient field arithmetic. Generalizations [2] and hardware architectures for such fields have also appeared in the literature [22,6,7]. Interestingly enough, all previous architectures have been optimized for either the time-area product metric or for performance (i.e. time) as these have been the parameters that designers traditionally look at. In this paper, we propose a somewhat different approach as area[3] is a fundamental constraint in both RFID systems and sensor networks. Thus, we revisit digit-serial multipliers and investigate ways to minimize their area requirements. Interestingly enough we come up with new digit-serial

[3] Power dissipation is also a key metric in RFID and sensor networks but in this paper we do not discuss it. However, notice that generally smaller area circuits tend to dissipate less power.

multipliers with reduced area requirements and improved critical path. Our key observation is that the modular reduction circuit can be placed in a different place in the multiplier giving a slight improvement in area and with a minor impact on the critical path of the architecture when compared to standard digit multipliers [25]. Compared to the work in [5], we improve both the area and the critical path.

The remainder of the paper is organized as follows. Section 2 gives a brief overview of related work. We briefly review digit multipliers and their complexity in Sect. 3. In Sect. 4, we introduce our new multiplier design aimed at reducing area, analyze its complexity and compare it to the architecture of [5], which is the smallest multiplier design known in the literature while still achieving acceptable performance. Finally, in Sect. 5 we provide conclusions and point out future work.

2 Related Work

2.1 Previous Finite Field Multipliers

For fixed irreducible polynomials, there are three different types of architectures used to build $\mathbb{F}_{p^m}$ multipliers: array-, digit-, and parallel-multipliers [25]. Parallel multipliers process all coefficients of both operands at once and generate all the coefficients of the result at the same time. Parallel multipliers (see for example, [21,26,11]) have a high critical path delay but only require one clock cycle to complete a whole multiplication. Thus, parallel multipliers exhibit high throughput and they are best suited for applications requiring high speed. However, they are expensive in terms of area when compared to serial multipliers and most of the time prohibitive for cryptographic applications. Thus, they are not considered any further in this work. Array-type (or serial) multipliers process all the coefficients of the multiplicand in parallel in the first step, while the coefficients of the multiplier are processed serially. Array-type multiplication can be performed in two different ways, depending on the order in which the coefficients of the multiplier are processed: Least Significant Element (LSE) first multiplier and Most Significant Element (MSE) first multiplier. Digit-multipliers, introduced by Song and Parhi [25], are a generalization of serial multipliers which process more than one coefficient of the multiplicand at the time. They are also divided into Most Significant and Least Significant Digit Element first multipliers, depending on the order in which the coefficients of the polynomial are processed. Notice that if the digit size is set to one, a digit multiplier becomes a serial multiplier.

Kumar et al. [17] have recently presented optimum digit sizes for high speed implementations of Digit-serial multipliers. They show that digit sizes of the form $2^l - 1$, where l is an integer, have smaller critical path and area-time product. They also present different Digit-serial multiplier architectures based on the number of accumulators in the multiplier. Thus, the Double-Accumulator Multiplier (DAM) and N-Accumulator Multiplier (NAM) are described in [17]. They show that these multipliers provide different optimum choices for area-time product, enabling designers to choose the best optimum multiplier for a given timing requirement. Notice that additional accumulators in multipliers imply additional register resources and thus, higher area requirements than those of traditional digit multipliers.

For arbitrary irreducible polynomials, there is also a broad literature (see for example [23,15,10]). However, it is our opinion that for ultra-constrained environments such as RFID tags and sensor nodes, such methodologies are too expensive and provide a level of functionality that it is unnecessary for the application. Thus, we focus on fixed irreducible polynomial multipliers in the remainder of this paper.

Although small area architectures for finite field multipliers have not been extensively considered on their own, applications such as ECC for extremely constrained environments have been considered in the literature. Thus, we surveyed this literature as well.

2.2 ECC for Constrained Environments

Low-power and compact implementations became an important research area with the constant increase in the number of hand-held devices such as mobile phones, smart cards, PDAs *etc.* Schroeppel *et al.* [24] presented a design for ECC over binary fields that was optimized for power, space and time in order to provide digital signatures. The processor in [24] had an area complexity of 191,000 gates. From these, the multiplier over the composite field $\mathbb{F}_{2^{178}}$ required 6200 gates. The work of Goodman and Chandrakasan [13] also dealt with energy-efficient solutions. They proposed a domain-specific reconfigurable cryptographic processor (DSRCP) for ECC over prime and binary finite fields. No details are provided about the resources required by the multiplier. Özturk *et al.* [20] introduced modulus scaling techniques that are applicable for ECC over a prime field to develop a low-power elliptic curve processor architecture. They obtained an ECC processor over a 166-bit long prime of size 30,333 gates with a performance of 31.9 msec for point multiplication. In [12],

Gaubatz et al. compared implementations of Rabin's scheme and NTRU-Encrypt with an ECC solution for wireless sensor networks. The ECC processor occupied an area of 18,720 gates and used a prime field of order $\approx 2^{100}$.

RFID-based identification is an example of an emerging technology which requires authentication as a cryptographic service. Recently, Wolkerstorfer [27] showed that ECC based PKC is feasible on RFID-tags by implementing the ECDSA on a small IC. This work is the first complete ECC low-power and compact implementation that meets the constraints imposed by the EPC standard. Following, this line of work, Batina et al. [3,5,4] have also presented architectures for small arithmetic units suited for RFID and sensor networks. In [3,4], the authors investigate different trade-offs that can be made in an ALU suited for implementing ECC over $\mathbb{F}_{2^m}$. Their ALU consists of an MSD multiplier, a squarer and an adder. A 3-to-1 multiplexer allows to select the result depending on whether a multiplication, an addition, or a squaring operation is desired. The smallest design in [4] required 6306 gates for an ALU suited to $\mathbb{F}_{2^{131}}$. In [5], the authors proposed an ALU without a squarer. In particular, they combine a MSD multiplier with the ability to perform addition without a big hardware overhead. One key observation in this architecture is that multiplexers can be substituted for AND gates and flip-flops with a load input. We will discuss the design in more detail in Sect. 3.

3 Digit-Serial Multipliers for $\mathbb{F}_{p^m}$

3.1 Notation

In the following, we will consider the field $\mathbb{F}_{p^m}$ generated by an irreducible polynomial over $\mathbb{F}_p$ of degree m, $q(x) = x^m + Q(x) = x^m + \sum_{i=0}^{k} q_i x^i,\ k < m$. We assume α to be a root of $q(x)$, thus for $A, B, C \in \mathbb{F}_{p^m}$, we write $A = \sum_{i=0}^{m-1} a_i \alpha^i$, $B = \sum_{i=0}^{m-1} b_i \alpha^i$, $C = \sum_{i=0}^{m-1} c_i \alpha^i$, and $a_i, b_i, c_i \in \mathbb{F}_p$. Notice that by assumption $q(\alpha) = 0$ since α is a root of $q(x)$. Therefore,

$$\alpha^m = -Q(\alpha) = \sum_{i=0}^{k} (-q_i)\alpha^i \tag{1}$$

gives an easy way to perform modulo reduction whenever we encounter powers of α greater than $m-1$. Addition in $\mathbb{F}_{p^m}$ can be achieved as shown in (2)

$$C(\alpha) \equiv A(\alpha) + B(\alpha) = \sum_{i=0}^{m-1} (a_i + b_i)\alpha^i \tag{2}$$

where the addition $a_i + b_i$ is done in $\mathbb{F}_p$. Multiplication of two elements $A, B \in \mathbb{F}_{p^m}$ is written as $C(\alpha) = \sum_{i=0}^{m-1} c_i \alpha^i \equiv A(\alpha) \cdot B(\alpha)$, where the multiplication is understood to happen in the finite field $\mathbb{F}_{p^m}$ and all α^t, with $t \geq m$ can be reduced with (1). We will abuse our notation and throughout the text we will write $A \bmod q(\alpha)$ to mean *explicitly* the reduction step described previously. We will refer to A as the multiplicand and to B as the multiplier. Finally, notice that although our treatment is general (for any characteristic), we are particularly interested in binary fields for our particular application setting: ECC on ultra-constrained environments such as RFID and sensor nodes.

3.2 LSDE and MSDE Multipliers

Digit multipliers, introduced in [25] for fields $\mathbb{F}_{2^k}$, are a trade-off between speed, area, and power consumption. This is achieved by processing several of the multiplier's (B) coefficients at the same time. The number of coefficients that are processed in parallel is defined to be the digit-size and we denote it with the letter D.

For a digit-size D, we can denote by $d = \lceil m/D \rceil$ the total number of digits in a polynomial of degree $m-1$. Then, we can re-write the multiplier as $B = \sum_{i=0}^{d-1} B_i \alpha^{Di}$, where

$$B_i = \sum_{j=0}^{D-1} b_{Di+j} \alpha^j, \;\; 0 \leq i \leq d-1 \tag{3}$$

and we assume that B has been padded with zero coefficients such that $b_i = 0$ for $m-1 < i < d \cdot D$ (i.e. the size of B is $d \cdot D$ coefficients but $\deg(B) < m$). Hence,

$$C \equiv AB \bmod q(\alpha) = A \sum_{i=0}^{d-1} B_i \alpha^{Di} \bmod q(\alpha) \tag{4}$$

In the following, we revisit digit-serial/parallel multiplication algorithms. These algorithms are classified as Least Significant Digit-Element first multiplier (LSDE) and Most Significant Digit-Element first multiplier (MSDE). Here, we have used the word *element* as in [6] to clarify that the digits correspond to groups of $\mathbb{F}_p$ coefficients in contrast to the binary case where the digits are groups of bits [25]. Using (4), the product in this scheme can be calculated as follows

$$C \equiv AB \equiv [B_0 A + B_1 (A\alpha^D \bmod q(\alpha)) + \ldots + B_{d-1} (A\alpha^{D(d-2)} \alpha^D \bmod q(\alpha))] \bmod q(\alpha) \tag{5}$$

Algorithm 1 LSDE Multiplier

Require: $A = \sum_{i=0}^{m-1} a_i \alpha^i$, where $a_i \in \mathbb{F}_p$, $B = \sum_{i=0}^{\lceil \frac{m}{D} \rceil - 1} B_i \alpha^{Di}$, where B_i is as defined in (3)
Ensure: : $C \equiv A \cdot B = \sum_{i=0}^{m-1} c_i \alpha^i$, where $c_i \in \mathbb{F}_p$
1: $C \leftarrow 0$
2: **for** $i = 0$ to $\lceil \frac{m}{D} \rceil - 1$ **do**
3: $\quad C \leftarrow B_i A + C$
4: $\quad A \leftarrow A\alpha^D \bmod q(\alpha)$
5: **end for**
6: Return $(C \bmod q(\alpha))$

This is summarized in Algorithm 1. A similar derivation can be performed for MSDE multipliers resulting in Algorithm 2. Notice that in

Algorithm 2 MSDE Multiplier

Require: $A = \sum_{i=0}^{m-1} a_i \alpha^i$, where $a_i \in \mathbb{F}_p$, $B = \sum_{i=0}^{d-1} B_i \alpha^{Di}$, where B_i is as defined in (3)
Ensure: $C \equiv A \cdot B = \sum_{i=0}^{m-1} c_i \alpha^i$, where $c_i \in \mathbb{F}_p$
1: $C \leftarrow 0$
2: **for** $i = 0$ to $d - 1$ **do**
3: $\quad C \leftarrow AB_{d-1-i} + (C \bmod q(\alpha))\alpha^D$
4: **end for**
5: Return $(C \bmod q(\alpha))$

both LSDE and MSDE multipliers, the intermediate result C is of degree larger than $m = \deg(q(\alpha))$. This fact translates into two consequences: (i) both LSDE and MSDE multipliers require a number of storage elements which is larger than the degree of the irreducible polynomial $q(\alpha)$ and (ii) after d loop iterations, one must perform one extra reduction. In a LSDE multiplier, products of the form $W\alpha^D \bmod F(\alpha)$ occur (as seen in Step 4 and 3 of Algorithms 1 and 2, respectively) which have to be reduced. As in the serial multiplier case, one can derive equations for the modular reduction for *general* irreducible $F(\alpha)$ polynomials. However, as shown in [25,6], it is more interesting to search for polynomials that minimize the complexity of the reduction operation. We recall two theorems in [25] which define these optimum irreducible polynomials.

Theorem 1. *[25] Given an irreducible polynomial of the form* $q(\alpha) = \alpha^m + q_k\alpha^k + \sum_{j=0}^{k-1} q_j\alpha^j$, *with* $k < m$. *For* $t \leq m - 1 - k$, α^{m+t} *can be*

reduced to degree less than m in one step with the following equation:

$$\alpha^{m+t} \bmod q(\alpha) = -q_k \alpha^{k+t} - \sum_{j=0}^{k-1} q_j \alpha^{j+t} \tag{6}$$

Theorem 2. *[25] For digit multipliers with digit-element size D, when $D \leq m-k$, the intermediate results in Algorithms 1 and 2 can be reduced to degree less than m in one step.*

Theorems 1 and 2 implicitly say that for a given irreducible polynomial $q(\alpha) = \alpha^m + q_k \alpha^k + \sum_{j=0}^{k-1} q_j \alpha^j$, the digit-element size D has to be chosen based on the value of k, the second highest degree in the irreducible polynomial. The architectures of the LSDE and MSDE multipliers are shown in Fig. 1 and 2. From Fig. 1, we can identify the following components

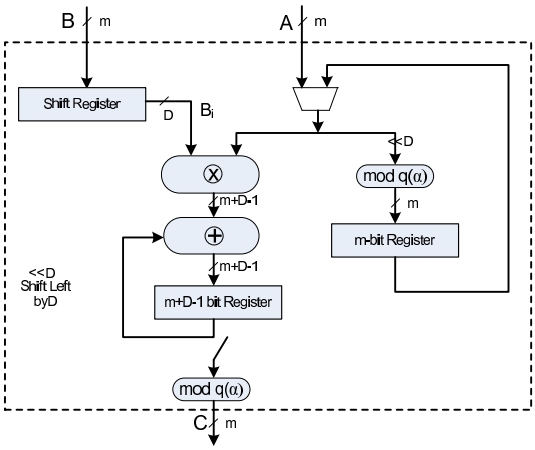

Fig. 1. $\mathbb{F}_{2^m}$ LSDE multiplier

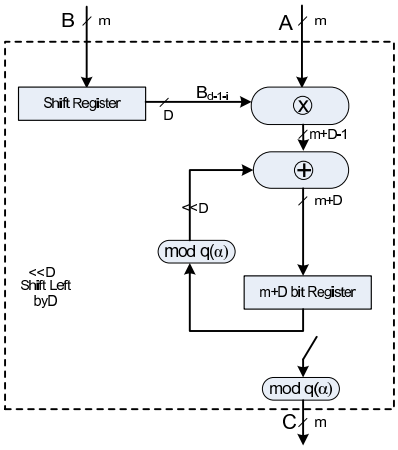

Fig. 2. $\mathbb{F}_{2^m}$ MSDE multiplier

for the LSDE multiplier:

1. The *multiplier core* which computes the intermediate $B_i A + C$ and stores it in the accumulator.
2. The *main reduction circuit* to shift A left by D and reduce the result $\bmod q(\alpha)$.
3. The *final reduction circuit* to reduce the contents in the accumulator to get the final result C.

4. A 2-to-1 m coefficient MUX which selects between the original input A and the updated A

Similarly, from Fig. 2, we can identify the following components:

1. The *multiplier core* which computes $B_i A + C$.
2. The *main reduction circuit* to shift the intermediate result C and reduce the result $\bmod q(\alpha)$.
3. The *final reduction circuit* to reduce the contents in the accumulator to get the final result C.

All the components run in parallel requiring one clock cycle to complete each step. The critical path of the whole multiplier normally depends on the critical path of the multiplier core. We notice that the design in [5] is also based on a MSDE multiplier (for binary fields). However, it has certain characteristics which makes it interesting and innovative. The authors are able to use an m coefficient (resp. bit) register instead of a $m+D-1$ coefficient register by performing a modular reduction for every coefficient (of the digit B_i) multiplication in the digit-multiplier core. The multiplier has a modular design and it is composed of identical cells. Each cell performs the operation $C^{(i)} \leftarrow \left(C^{(i-1)} + b_i A \bmod p(\alpha)\right) \alpha$. Thus, for a digit D they have D cells and D reduction circuits. This architecture is very similar to the folded MSDE multiplier of [25]. Batina et al. further minimize the size of the overall ALU by allowing the accumulator register C to have a load input and taking advantage of the fact that if C is loaded with an initial value I, then we obtain an adder for free without extra hardware[4].

We do not provide here an analysis of the area requirements and the critical path of the different components of the multiplier. Rather, we refer to [25,14,17] for such an analysis. Table 1 summarizes the area requirements and time complexity of known $\mathbb{F}_{2^m}$ digit multipliers. We estimate the area in terms of AND gates, XOR gates, single-bit Flip-Flops, and (2:1)-bit MUXes. The number of XORs and the critical path is based on the assumption that a binary tree structure is used to add the required elements and that a known fixed irreducible polynomial is used (thus, the modular reduction circuit is composed of only XOR gates). For n elements, the number of XOR gates required is $n-1$ and the critical path delay becomes the binary tree depth $\lceil \log_2 n \rceil$. We calculate the critical path as a function of the delay of XORs (Δ_{XOR}), AND gates (Δ_{AND}),

[4] The observation that we can calculate $A \cdot B + I$ for "free" first appears in [19] in the context of ECC processors.

and MUXes (Δ_{MUX}). This allows our analysis to be independent of the cell-technology and field used for the implementation. Notice that we do not make any assumptions about the form of the irreducible polynomial (whether it is a trinomial, pentanomial, etc.) chosen to perform the reduction operation. However, we do assume that the irreducible polynomial is known ahead of time, implying that the modular reduction operation requires only XORs. This also implies that these complexities are upper bounds. We do not include the multiplier of [17] in Table 1 because it requires more hardware resources than those presented by others.

Table 1. Area complexity and critical path delay of digit multipliers over $\mathbb{F}_{2^m}$. Digit size D and fixed $p(\alpha)$ satisfying Theorem 1.

Multiplier Type	Circuit	Area Complexity				Critical Path Delay	Latency (# clocks)
		AND	XOR	MUX	FF		
LSDE [25]	Shift-register B	–	–	–	m	$\lceil \log_2(D+1) \rceil$ $\Delta_{XOR}+$ $\Delta_{AND}+$ Δ_{MUX}	$\lceil m/D \rceil + 1 - \delta(1, D)$
	$AB_i + C$	mD	mD	–	–		
	$A \bmod q(\alpha)$	–	kD	m	m		
	Accumulator C	–	–	–	$m + D - 1$		
	Final reduction	–	$(k+1)(D-1)$	–	–		
	Overall	mD	$D(m + 2k + 1) - (k + 1)$	m	$3m + D - 1$		
MSDE [25]	Shift-register B	–	–	–	m	$\lceil \log_2(2D+1) \rceil$ $\Delta_{XOR}+$ Δ_{AND}	$\lceil m/D \rceil + 1 - \delta(1, D)$
	$AB_{d-1-i} + (C \bmod q(\alpha))\alpha^D$	mD	$(m + k)D$	–	–		
	Accumulator C	–	–	–	$(m + D)$		
	Final reduction	–	$(k+1)(D-1)$	–	–		
	Overall	mD	$D(m + 2k + 1) - (k + 1)$		$2m + D$		
MSDE [5]	Shift-register B	–	–	–	m	$2D\Delta_{XOR}+$ Δ_{AND}	$\lceil m/D \rceil$
	Accumulator C	–	–	–	m		
	D cells computing $b_j A\alpha \bmod p(\alpha)$	$mD + D$	$D(m + k + 1)$	–	–		
	Overall	$D(m + 1)$	$D(m + k + 1)$	–	$2m$		

4 Improved Digit Multipliers

4.1 Idea

We recall the Step 3 of Algorithm 2 in (7), which is the step that is performed in every iteration of the algorithm.

$$C \leftarrow AB_{d-1-i} + (C \bmod q(\alpha))\alpha^D \tag{7}$$

We notice that (7) can be re-written as (8), where we have made used of the fact that the modular reduction operation is commutative with addition and multiplication operations in the field of definition.

$$C \leftarrow \left(AB_{d-1-i} + C\alpha^D\right) \bmod q(\alpha) \tag{8}$$

Algorithm 3 Modified MSDE Multiplier

Require: $A = \sum_{i=0}^{m-1} a_i \alpha^i$, where $a_i \in \mathbb{F}_p$, $B = \sum_{i=0}^{d-1} B_i \alpha^{Di}$, where B_i is as defined in (3)
Ensure: $C \equiv A \cdot B = \sum_{i=0}^{m-1} c_i \alpha^i$, where $c_i \in \mathbb{F}_p$
1: $C \leftarrow 0$
2: **for** $i = 0$ to $d-1$ **do**
3: $\quad C \leftarrow \left(AB_{d-1-i} + C\alpha^D\right) \bmod q(\alpha)$
4: **end for**

Using (8), we can re-write Algorithm 2 as Algorithm 3. Notice that now $\deg(C) < m$, in every iteration of the algorithm. Algorithm 3 implies the multiplier shown in Figure 3 for the particular case of a binary field. Notice that Algorithm 3 does not require the last modular reduction in

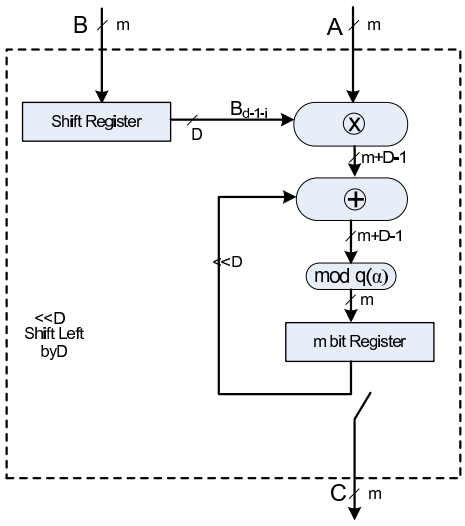

Fig. 3. Modified MSDE multiplier in $\mathbb{F}_{2^m}$

Algorithm 2 thus resulting in an area complexity advantage. However, when compared to the work in [5], it is not clear if there are any advantages. Thus, we analyze the complexity of the multiplier next.

4.2 Area-Time Complexity for Optimal Irreducible Polynomials over $\mathbb{F}_{2^m}$

Before estimating the complexity of the modified MSDE multiplier, it is helpful to obtain equations to describe the values of C at iteration i in Algorithm 3. Thus, assume that B_i is as in (3), $q(\alpha) = \alpha^m + \sum_{s=0}^{k} q_s \alpha^s$ (Theorem 1), $A = \sum_{i=0}^{m-1} a_i \alpha^i$, and $D \leq m - k$ (Theorem 2). We also define intermediate variables C' and E. Then,

$$\begin{aligned} C'^{(i)} &= \sum_{j=0}^{m+D-1} c_j'^{(i)} \alpha^j = E^{(i)} + C^{(i-1)} \alpha^D \\ &= c_{m-1}^{(i-1)} \alpha^{m+D-1} + \sum_{j=D}^{m+D-2} \left(e_j^{(i)} + c_{j-D}^{(i-1)} \right) \alpha^j + \sum_{j=0}^{D-1} e_j^{(i)} \alpha^j \qquad (9) \end{aligned}$$

$$C^{(i)} = C'^{(i)} \bmod p(\alpha) = \sum_{j=0}^{m-1} c_j'^{(i)} \alpha^j + \sum_{s=0}^{k} \sum_{j=0}^{D-1} \left(-q_s \cdot c_{j+m}'^{(i)} \right) \alpha^{j+s} \qquad (10)$$

where $C^{(-1)} = 0$ and

$$\begin{aligned} E^{(i)} &= \sum_{j=0}^{m+D-2} e_j^{(i)} \alpha^j = B_{d-1-i} \cdot A = \left(\sum_{j=0}^{m-1} a_j \alpha^j \right) \left(\sum_{s=0}^{D-1} b_{D(d-1-i)+s} \alpha^s \right) \\ &= \sum_{j=0}^{m-1} \sum_{s=0}^{D-1} \left(a_j \cdot b_{D(d-1-i)+s} \right) \alpha^{j+s} \qquad (11) \end{aligned}$$

For the particular case of $\mathbb{F}_{2^m}$, the above implies the following:

- It follows from (9) that to compute the intermediate result C' we require $m - 1$ XOR gates.
- It follows from (11) that in each iteration one requires mD AND gates in parallel and $\sum_{j=0}^{D-2} j + \sum_{j=D-1}^{m-1} (D-1) + \sum_{j=m}^{m+D-2} (m+D-2-j) = (D-1)(m-1)$ XOR gates to compute $E^{(i)}$.
- The previous statement also implies that the double summation in (10) requires $m(k+1)$ AND gates and $(D-1)k$ XOR gates. However, assuming that $p(\alpha)$ is fixed and known ahead of time, the AND gates are not required and we are only left with the XOR gates. Thus, the overall complexity of (10) is $(D-1)k + (k+D) = D(k+1)$ XOR gates.
- To reduce the critical path of the multiplier we also use a binary tree architecture. We have D terms resulting from the modular reduction

and $D+1$ from the multiplier and the accumulator. This results in a maximal critical path of $(\lceil \log_2(D) \rceil + \lceil \log_2(D+1) \rceil)\, \Delta_{XOR} + \Delta_{AND}$.

The previous discussion is summarized in Table 2.

Table 2. Area complexity and critical path delay of modified digit multiplier over $\mathbb{F}_{2^m}$. Digit size D and fixed $p(\alpha)$ satisfying Theorem 1.

Multiplier Type	Circuit	Area Complexity				Critical Path Delay	Latency (# clocks)
		AND	XOR	MUX	FF		
modified MSDE	Shift-register B	–	–	–	m	$(\lceil \log_2(D) \rceil + \lceil \log_2(D+1) \rceil)\, \Delta_{XOR} + \Delta_{AND}$	$\lceil m/D \rceil$
	$(AB_{d-1-i} + C\alpha^D) \bmod q(\alpha)$	mD	$(m+k)D$	–	–		
	Accumulator C	–	–	–	m		
	Overall	mD	$(m+k)D$		$2m$		

From Table 2 and 1, it is easy to see that the proposed multiplier results in a small area improvement when compared to [5] and in a significant improvement in terms of critical path.

4.3 Other Observations

We notice that the present architecture can be combined with some of the design choices made by Batina et al. [5]. In particular, the idea of providing a loadable C register can be combined with the present multiplier as well. This would lead to a somewhat smaller implementation of an ALU targeting an ECC implementation.

Notice also that the same idea used in [5] to avoid having to implement an adder can be applied to LSDE multipliers to reduce their area complexity. In particular, not only can you provide a loadable C register but also a loadable A register. This would allow the designer get rid off the MUX in Fig. 1. Nevertheless, it is apparent from the results that MSDE architectures are always better in terms of area when compared to LSDE-based multipliers.

5 Concluding Remarks

This work introduces a new finite field multiplier based on a modification of the standard MSDE multiplier introduced in [25]. The modification results in a slight improvement in the area complexity of the multiplier when compared to [5] and in the ability to reduce the critical path of the multiplier from linear complexity to logarithmic complexity. We conclude

that the presented design should be the architecture of choice for implementations of ECC that target ultra-constrained environments such as RFID tags and sensor nodes. We also notice that the present architecture shows even greater appeal for applications in which the field characteristic is non-binary, such as systems based on the computation of the Tate pairing. Future work will include the implementation of the multiplier in hardware and its application to an ECC processor.

References

1. D. V. Bailey and C. Paar. Efficient Arithmetic in Finite Field Extensions with Application in Elliptic Curve Cryptography. *Journal of Cryptology*, 14(3):153–176, 2001.
2. S. Baktir and B. Sunar. Optimal tower fields. *IEEE Trans. Computers*, 53(10):1231–1243, 2004.
3. L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. An Elliptic Curve Processor Suitable For RFID-Tags. Cryptology ePrint Archive, Report 2006/227, July 4th, 2006. Available at `http://eprint.iacr.org/`.
4. L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede. Public-Key Cryptography for RFID-Tags. In *IEEE Conference on Pervasive Computing and Communications Workshops — PerCom 2007 Workshops*, New York, March 19-23, 2007. IEEE Computer Society.
5. L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede. Low-cost Elliptic Curve Cryptography for Wireless Sensor Networks. In L. Buttyan, V. Gligor, and D. Westhoff, editors, *European Workshop on Security and Privacy in Ad hoc and Sensor Networks — ESAS 2006*, volume 4357 of *LNCS*, pages 6–17. Springer, September 20-21, 2006.
6. G. Bertoni, J. Guajardo, S. S. Kumar, G. Orlando, C. Paar, and T. J. Wollinger. Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications. In M. Joye, editor, *Topics in Cryptology - CT-RSA 2003*, volume 2612 of *LNCS*, pages 158–175. Springer, April 13-17, 2003.
7. G. Bertoni, J. Guajardo, and G. Orlando. Systolic and Scalable Architectures for Digit-Serial Multiplication in Fields $GF(p^m)$. In T. Johansson and S. Maitra, editors, *Progress in Cryptology - INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 349–362. Springer, December 8-10, 2003.
8. H. Chan and A. Perrig. Security and Privacy in Sensor Networks. *IEEE Computer*, 36(10):103–105, 2003.
9. D. E. Culler, D. Estrin, and M. B. Srivastava. Guest editors' introduction: Overview of sensor networks. *IEEE Computer*, 37(8):41–49, 2004.
10. H. Fan and M. A. Hasan. Relationship between $GF(2^m)$ Montgomery and Shifted Polynomial Basis Multiplication Algorithms. *IEEE Trans. Computers*, 55(9):1202–1206, 2006.
11. H. Fan and M. A. Hasan. A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields. *IEEE Trans. Computers*, 56(2):224–233, 2007.
12. G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar. State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks. In *2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005)*, Kauai Island, Hawaii, March 2005.

13. J. Goodman and A.P. Chandrakasan. An energy-efficient reconfigurable public-key cryptography processor. *IEEE Journal of Solid-State Circuits*, 36(11):1808–1820, November 2001.
14. J. Guajardo Merchan. *Arithmetic Architectures for Finite Fields $GF(p^m)$ with Cryptographic Applications*. PhD thesis, Fakultät für Elektrotechnik und Informationstechnik, Ruhr-Universität-Bochum, Bochum, Germany, July 2004. Available from `http://www.crypto.rub.de`.
15. A. A.-A. Gutub, A. F. Tenca, E. Savas, and Ç. K. Koç. Scalable and Unified Hardware to Compute Montgomery Inverse in $GF(p)$ and $GF(2)$. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume 2523 of *LNCS*, pages 484–499. Springer, August 13-15, 2002.
16. A. Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, February 2006. Extended version available from `http://www.rsasecurity.com/rsalabs/node.asp?id=2029`.
17. S. Kumar, T. J. Wollinger, and C. Paar. Optimum Digit Serial $GF(2^m)$ Multipliers for Curve-Based Cryptography. *IEEE Trans. Computers*, 55(10):1306–1311, 2006.
18. P. Mihăilescu. Optimal Galois Field Bases which are not Normal. Recent Results Session — FSE '97, 1997.
19. G. Orlando and C. Paar. A High Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, volume 1965 of *LNCS*, pages 41–56. Springer, August 17-18, 2000.
20. E. Özturk, B. Sunar, and E. Savaş. Low-Power Elliptic Curve Cryptography Using Scaled Modular Arithmetic. In M. Joye and J. J. Quisquater, editors, *Cryptographic Hardware in Embedded Systems — CHES 2004*, volume 3156 of *LNCS*, pages 92–106. Springer-Verlag, 2004.
21. C. Paar. A New Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Fields. *IEEE Trans. Computers*, 45(7):856–861, 1996.
22. D. Page and N. P. Smart. Hardware implementation of finite fields of characteristic three. In B. S. Kaliski, Jr., Ç. K. Koç, and C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2002*, volume LNCS 2523, pages 529–539. Springer-Verlag, 2002.
23. E. Savas, A. F. Tenca, and Ç. K. Koç. A Scalable and Unified Multiplier Architecture for Finite Fields $GF(p)$ and $GF(2^m)$. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, volume 1965 of *LNCS*, pages 277–292. Springer, August 17-18, 2000.
24. R. Schroeppel, C. L. Beaver, R. Gonzales, R. Miller, and T. Draelos. A Low-Power Design for an Elliptic Curve Digital Signature Chip. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume 2523 of *LNCS*, pages 366–380, 2002.
25. L. Song and K. K. Parhi. Low Energy Digit-Serial/Parallel Finite Field Multipliers. *Journal of VLSI Signal Processing*, 19(2):149–166, June 1998.
26. B. Sunar. A Generalized Method for Constructing Subquadratic Complexity $GF(2^k)$ Multipliers. *IEEE Trans. Computers*, 53(9):1097–1105, 2004.
27. J. Wolkerstorfer. Scaling ECC Hardware to a Minimum. In ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005, September 6-7 2005. Invited talk.