Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications

Guido Bertoni[1], Jorge Guajardo[2], Sandeep Kumar[2], Gerardo Orlando[3], Christof Paar[2], and Thomas Wollinger[2]

[1] Politecnico di Milano, P.za L. Da Vinci 32
20133 Milano, Italy
bertoni@elet.polimi.it
[2] Communication Security Group, Ruhr-Universität Bochum
44780 Bochum, Germany
{guajardo,sandeep,cpaar,wollinger}@crypto.rub.de
[3] General Dynamics Communication Systems
77 A St. Needham MA 02494, USA
Gerardo.Orlando@GDC4S.Com

Abstract. Recently, there has been a lot of interest on cryptographic applications based on fields $GF(p^m)$, for $p>2$. This contribution presents $GF(p^m)$ multipliers architectures, where p is odd. We present designs which trade area for performance based on the number of coefficients that the multiplier processes at one time. Families of irreducible polynomials are introduced to reduce the complexity of the modulo reduction operation and, thus, improved the efficiency of the multiplier. We, then, specialize to fields $GF(3^m)$ and provide the first cubing architecture presented in the literature. We synthesize our architectures for the special case of $GF(3^{97})$ on the XCV1000-8-FG1156 and XC2VP20-7-FF1156 FPGAs and provide area/performance numbers and comparisons to previous $GF(3^m)$ and $GF(2^m)$ implementations. Finally, we provide tables of irreducible polynomials over $GF(3)$ of degree m with $2 \leq m \leq 255$.

1 Introduction

Galois field arithmetic has received considerable attention in recent years due to their application in public-key cryptography schemes and error correcting codes. In particular, two public-key cryptosystems based on finite fields stand out: elliptic curve (EC) cryptosystems, introduced by Miller and Koblitz [24, 19], and hyperelliptic cryptosystems, a generalization of elliptic curves introduced by Koblitz in [20]. Both, prime fields and extension fields, have been proposed for use in such cryptographic systems. However, until a few years ago the focus was mainly on fields of characteristic 2 due to the straight forward manner in which elements of $GF(2)$ can be represented, i.e., they can be represented by the logical values "0" and "1". For these types of fields, both software implementations and hardware architectures have been studied extensively. In recent years, $GF(p^m)$ fields, where p is odd, have gained interest in the research community.

M. Joye (Ed.): CT-RSA 2003, LNCS 2612, pp. 158–175, 2003.

Mihălescu [28] and independently Bailey and Paar [3, 4] introduced the concept of Optimal Extension Fields (OEFs) in the context of elliptic curve cryptography. OEFs are fields $GF(p^m)$ 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. The treatment in [4, 28] and that of other works based on OEFs has only been concerned with efficient software implementations.

In [21, 32], $GF(p^m)$ fields are proposed for cryptographic purposes where p is relatively small. [21] describes an implementation of ECDSA over fields of characteristic 3 and 7. The author in [32] describes a method to implement elliptic curve cryptosystems over fields of *small* odd characteristic, only considering $p < 24$ in the results section. More recently, Boneh and Franklin [8] introduced an identity-based encryption scheme based on the use of the Weil and Tate pairings. Similarly, [7] described a short signature scheme based on the Weil and Tate pairings. Other applications include [17, 34]. All of these applications consider elliptic curves defined over fields of characteristic 2 and 3. Because characteristic 2 field arithmetic has been extensively studied in the literature, authors have concentrated the efforts to improve the performance of systems based on characteristic 3 arithmetic[1]. For example, [5, 10] describe algorithms to improve the efficiency of the pairing computations. [10] also introduces some clever tricks to improve the efficiency of the underlying arithmetic in *software* based solutions. [29] treats the hardware implementation of fields of characteristic 3. Their design is only geared towards fields of characteristic 3. Moreover, it is acknowledged that the element representation makes their architectures unsuitable to take advantage of operations such as cubing (i.e. a cubing will only be as fast as a general multiplication, while in other implementations cubing could be more efficient) which are important in Tate pairing computations.

1.1 Our Contributions

Given the research community's interest on cryptographic systems based on fields of odd characteristic and the lack of hardware architectures for general odd characteristic fields, we try to close this gap. Our approach is different from previous ones, in that, we propose *general* architectures which are suitable for fields $GF(p^m)$ with p odd. In particular, we generalize the work in [33] to fields $GF(p^m)$, p odd. We, then, study carefully the case of $GF(3^m)$ due to its cryptographic significance. In addition, we focused on finding irreducible polynomials over $GF(3)$ to improve the performance of the multiplier. For the problem of efficient $GF(p)$ arithmetic, we refer the reader to [9, 30, 13].

The remaining of this contribution is organized as follows. In Section 2 we survey previous $GF(p^m)$ architectures and discuss certain multiplier architectures for $GF(2^k)$ type fields, which we generalize for the $GF(p^m)$ case in Section 4. In

[1] The use of characteristic 3 fields is preferred in some applications due to the improved bandwidth requirements implied by the security parameters. For example, signatures resulting from Pairing cryptography will be smaller in characteristic 3 than in characteristic 2.

Section 4, we also study both the time and area complexities of the multipliers and give two theorems which help us in choosing irreducible polynomials that will minimize the area and delay of the resulting multipliers. Finally, Section 5 specializes the results of the previous section to the case of $GF(3^m)$ fields which recently have become of great interest in the research community. We also describe a prototype implementation of our architectures for three different fields on two FPGAs. We also provide tables of irreducible polynomials over $GF(3)$ for degrees between 2 and 255. We end this contribution with some conclusions.

2 Related Work

In contrast to the $GF(p)$ case, there has not been a lot of work done on $GF(p^m)$ architectures. Our literature search yielded [31] as the only reference that explicitly treated the general case of $GF(p^m)$ multipliers, p odd[2]. In [31], $GF(p^m)$ multiplication is computed in two stages. First the polynomial product is computed modulo a highly factorizable degree S polynomial, $M(x)$, with $S \geq 2m-1$. The product is, thus, computed using a polynomial residue number system. The second step involves reducing modulo the irreducible polynomial $p(x)$ over which $GF(p^m)$ is defined. The method, however, does not seem to apply to field sizes common in cryptographic applications due to certain constraints on the size of m. In particular, for $p = 3$, there does not exist a suitable $M(x)$ polynomial.

The authors in [25] consider multiplier architectures for composite fields of the form $GF((3^n)^3)$ using Multi-Value Logic (MVL) and a modified version of the Karatsuba algorithm [18] for polynomial multiplication over $GF((3^n)^3)$. Elements of $GF((3^n)^3)$ are represented as polynomials of maximum degree 2 with coefficients in $GF(3^n)$. Multiplication in $GF(3^n)$ is achieved in the obvious way. Karatsuba multiplication is combined with modular reduction over $GF((3^n)^m)$ to reduce the complexity of their design. Because of the use of MVL no discussion of modulo 3 arithmetic is given. The authors estimate the complexity of their design for arithmetic over $GF((3^2)^3)$ as 56 mod-3 adders and 67 mod-3 multipliers. To our knowledge, [29] is the first work that describes $GF(3^m)$ architectures for applications of cryptographic significance, thus we describe it in some detail. The authors describe a representation similar to the one used by [10] to represent their polynomials. They combine all the least significant bits of the coefficients of an element, say A, into one value and all the most significant bits of the coefficients of A into a second value (notice the coefficients of A are elements of $GF(3)$ and thus 2 bits are needed to represent each of them). Thus, $A = (a_1, a_0)$ where a_1 and a_0 are m-bit long each. Addition of two polynomials $A = (a_1, a_0)$, $B = (b_1, b_0)$ with $C = (c_1, c_0) \equiv A + B$ is achieved as:

$$t = (a_1 \vee b_0) \oplus (a_0 \vee b_1), \quad c_1 = (a_0 \vee b_0) \oplus t, \quad c_2 = (a_1 \vee b_1) \oplus t \qquad (1)$$

where $\vee$ and $\oplus$ mean the logical OR and exclusive OR operations, respectively. The authors of [29] notice that subtraction and multiplication by 2 are equiva-

[2] There has been a lot work done, however, on finite field architectures for characteristic two fields.

lent in characteristic 3 and that they can be achieved as $2 \cdot A = 2 \cdot (a1, a0) = -A = -(a1, a0) = (a0, a1)$. Multiplication is achieved in the bit-serial manner, by repeatedly shifting the multiplier down by one bit position and shifting the multiplicand up by one bit position. The multiplicand is then added or subtracted depending on whether the least significant bit of the first or second word of the multiplier is equal to one. They notice that with this representation a cubing operation is only as fast as a general multiply, whereas, using other implementation methods the cubing operation is much faster. Finally, [29] also discuss the implementation of multiplication in $GF((3^m)^6)$ using the irreducible polynomial $Q(y) = y^6 + y + 2$. They use the school book method to multiply polynomials of degree 5 with coefficients in $GF(3^m)$ and then reduce modulo $Q(y)$ using 10 additions and 4 doublings in $GF(3^m)$. They provide timings which we further discuss in Section 6.2.

In [33] a new approach for the design of digit-serial/parallel $GF(2^k)$ multipliers is introduced. Their approach combines both array-type and parallel multiplication algorithms, where the digit-type algorithms minimize the latency for one multiplication at the expense of extra hardware inside each digit cell. In addition, the authors consider special types of polynomials which allow for efficiency in the modulo $p(x)$ reduction operation. These architectures are generalized in Section 4 to the $GF(p^m)$ case, where p is odd.

3 Mathematical Background

For a thorough introduction to finite fields, we refer the reader to [22]. Here, we briefly review the theory that we will need to develop the architectures of this paper. In the following, we will consider the field $GF(p^m)$ generated by an irreducible polynomial $p(x) = x^m + P(x) = x^m + \sum_{i=0}^{m-1} p_i x^i$ over $GF(p)$ of degree m. Let α be a root of $p(x)$, then we can represent $A \in GF(p^m)$ in polynomial basis as $A(\alpha) = \sum_{i=0}^{m-1} a_i \alpha^i$, $a_i \in GF(p)$. Notice that by assumption $p(\alpha) = 0$ since α is a root of $p(x)$. Therefore,

$$\alpha^m = -P(\alpha) = \sum_{i=0}^{m-1} (-p_i)\alpha^i \tag{2}$$

gives an easy way to perform modulo reduction whenever we encounter powers of α greater than $m - 1$.

In what follows, it is assumed that $A, B, C \in GF(p^m)$, with $A = \sum_{i=0}^{m-1} a_i \alpha^i$, $B = \sum_{i=0}^{m-1} b_i \alpha^i$, $C = \sum_{i=0}^{m-1} c_i \alpha^i$, and $a_i, b_i, c_i \in GF(p)$. Then, addition in $GF(p^m)$ can be achieved as shown in (3)

$$C(\alpha) \equiv A(\alpha) + B(\alpha) = \sum_{i=0}^{m-1} (a_i + b_i)\alpha^i \tag{3}$$

where the addition $a_i + b_i$ is done in $GF(p)$. We write the multiplication of two elements $A, B \in GF(p^m)$ 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 $GF(p^m)$ and all α^t, with $t \geq m$ can be reduced with (2). Much of the remaining of this paper is devoted to efficient ways to perform multiplication in hardware. We abuse our notation and throughout the text we will write $A \bmod p(\alpha)$ to mean *explicitly* the reduction step described previously. Finally, we refer to A as the multiplicand and to B as the multiplier.

4 General Architectures for $GF(p^m)$ Arithmetic

This section is concerned with hardware architectures for addition and multiplication in $GF(p^m)$. Inversion can be performed through the Euclidean algorithm or by exponentiation based techniques (see for example [12]) and we do not treat it any further in this paper.

4.1 Adders

Addition in $GF(p^m)$ is performed according to (3). A parallel adder requires m $GF(p)$ adders and its critical path delay is one $GF(p)$ adder.

4.2 Multipliers

There are three different types of architectures used to build $GF(p^m)$ multipliers: array-, digit-, and parallel-multipliers [33]. 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 not described here because of lack of space. Digit-multipliers are also divided in Most Significant and Least Significant Digit-Element first multipliers, depending on the order in which the coefficients of the polynomial are processed. Parallel-multipliers 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 and relatively small finite fields. However, they are expensive in terms of area when compared to serial multipliers and thus most of the time prohibitive for cryptographic applications. We don't discuss parallel-multipliers any further in this paper.

Least Significant Element (LSE) First Multiplier. The LSE scheme processes first coefficient b_0 of the multiplier and continues with the remaining coefficients one at the time in ascending order. Hence, multiplication according to this scheme can be performed in the following way:

$$C \equiv AB \bmod p(\alpha) \equiv b_0 A + b_1(A\alpha \bmod p(\alpha)) + \ldots + b_{m-1}(A\alpha^{m-1} \bmod p(\alpha))$$

Algorithm 1 LSE Multiplier

Require: $A = \sum_{i=0}^{m-1} a_i \alpha^i$, $B = \sum_{i=0}^{m-1} b_i \alpha^i$, where $a_i, b_i \in GF(p)$
Ensure: $C \equiv A \cdot B = \sum_{i=0}^{m-1} c_i \alpha^i$, where $c_i \in GF(p)$
  $C \leftarrow 0$
  for $i = 0$ to $m-1$ **do**
    $C \leftarrow b_i A + C$
    $A \leftarrow A\ \alpha \bmod p(\alpha)$
  end for
  Return (C)

The accumulation of the partial product has to be done with a polynomial adder. This multiplier computes the operation according to Algorithm 1. The adapted substructure sharing technique [16] can be used to compute the LSE multiplication in a more efficient way, if the LSE multiplier is used as building block for a larger system, like a system with broadcast structure.

Reduction mod $p(\alpha)$. In LSE multipliers the quantity $W\alpha$, where $W(\alpha) = \sum_{i=0}^{m-1} w_i \alpha^i \in GF(p^m)$, has to be reduced mod$p(\alpha)$. Multiplying W by α,

$$W\alpha = \sum_{i=0}^{m-1} w_i \alpha^{i+1} = w_{m-1}\alpha^m + \sum_{i=0}^{m-2} w_i \alpha^{i+1}$$

Using (2) and re-writing the index of the second summation, $W\alpha \bmod p(\alpha)$ can then be calculated as follows:

$$W\alpha \bmod p(\alpha) \equiv (-p_0 w_{m-1}) + \sum_{i=1}^{m-1} (w_{i-1} - p_i w_{m-1})\alpha^i \tag{4}$$

where all coefficient arithmetic is done modulo p. Using (4) we can write expressions for A and C in Algorithm 1 at iteration i as follows:

$$C^{(i)} = \sum_{j=0}^{m-1} c_j^{(i)} \alpha^j \equiv b_i A^{(i)} + C^{(i-1)} = \sum_{j=0}^{m-1} (b_i a_j^{(i)} + c_j^{(i-1)})\alpha^j,$$

$$A^{(i)} = \sum_{j=0}^{m-1} a_j^{(i)} \alpha^j \equiv A^{(i-1)}\alpha \equiv (-p_0 a_{m-1}^{(i-1)}) + \sum_{j=1}^{m-1} (a_{j-1}^{(i-1)} - p_j a_{m-1}^{(i-1)})\alpha^j$$

with $C^{(-1)} = 0$ and $A^{(-1)} = A$. As a final remark, notice that if you initialize C to a value different from 0, say I, then Algorithm 1 computes $C \equiv A \cdot B + I \bmod p(\alpha)$. This multiply-accumulate operation turns out to be very useful in elliptic curve systems and it is obtained at no extra cost.

Table 1. Area complexity and critical path delay of LSE multiplier

Irreducible polynomial	Area Complexity	Critical Path Delay	Latency (# clocks)
r-nomial	$(m+r-2)$ ADD + $(m+r-1)$ MUL	1 ADD + 1 MUL	m
General	$(2m-1)$ ADD + $2m$ MUL	1 ADD + 1 MUL	m

Area/Time Complexity of LSE Multipliers. LSE multipliers take m iterations to output the product $C \equiv A \cdot B \bmod p(\alpha)$. In each iteration, the following operations are performed: 1 multiplication of a $GF(p)$ element by a $GF(p^m)$ element (requires m $GF(p)$ multipliers), 1 $GF(p^m)$ addition (requires m $GF(p)$ adders), 1 multiplication by α (implemented as a $GF(p)$ coefficient shift), and 1 modulo $p(\alpha)$ reduction. This last operation could be implemented according to (4), and thus, it would require $(r-1)$ $GF(p)$ multipliers and $(r-2)$ adders for a fixed r-nomial (where r is the number of non-zero coefficients in the irreducible polynomial $p(x)$). However, it could be desired to load the modulus on demand, in which case one would need m $GF(p)$ multipliers and $(m-1)$ adders.

Both the area and time complexities of Algorithm 1 are summarized in Table 1 in terms of $GF(p)$ adders and multipliers, for two types of irreducible polynomials. This unusual measure is independent of technology and thus most general. One immediate advantage of estimating area in terms of $GF(p)$ adders (multipliers) is that we don't need to care about the way these are implemented. In particular, there are many implementation choices depending on your application and design criteria [9, 13, 30]. Section 6.1 gives specific complexity numbers for an FPGA implementation of $GF(3^m)$ arithmetic in terms of both Look-Up Tables[3] (LUTs), also known as Configurable Logic Blocks (CLBs), and flip-flops (FF), thus taking into account the way $GF(p)$ arithmetic is implemented on the target technology. In Table 1, ADD and MUL refer to the area and delay of a $GF(p)$ adder and multiplier, respectively. We have not taken into account the delays or area requirements of storage elements (such as those needed to implement a shift register) or routing elements (such as those used for interconnections in FPGAs). In addition, we do not make any distinction between general and constant $GF(p)$ multipliers, i.e., we assume their complexities are the same. Finally, general irreducible polynomials refer to the case in which you want to be able to change the irreducible polynomial on demand.

Digit-Serial/Parallel Multipliers. LSE multipliers process the coefficients of A in parallel, while the coefficients of B are processed serially. Hence, these multipliers are area-efficient and suitable for low-speed applications. Digit multipliers, introduced in [33] for fields $GF(2^k)$, are a trade-off between speed, area, and power consumption. This is achieved by processing several of B's 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 by D. For a digit-size D, we can denote

[3] Look-Up Tables are the basic building blocks of most common FPGAs [1, 2, 35].

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

$$B_i = \sum_{j=0}^{D-1} b_{Di+j} \alpha^j \quad 0 \leq i \leq d-1 \tag{5}$$

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. B's size is $d \cdot D$ coefficients but $\deg(B) < m$). Hence,

$$C \equiv AB \bmod p(\alpha) = A \sum_{i=0}^{d-1} B_i \alpha^{Di} \bmod p(\alpha) \tag{6}$$

In the following, a generalized digit-serial/parallel multiplication algorithm is introduced. We named this algorithm Least Significant Digit-Element first multiplier (LSDE), where the word *element* was introduced to clarify that the digits correspond to groups of $GF(p)$ coefficients in contrast to [33] where the digits where groups of bits. The LSDE is an extension of the LSE multiplier. Using (6), the product $C \equiv AB \bmod p(\alpha)$ in this scheme can be calculated as follows

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

This is summarized in Algorithm 2. The adapted substructure sharing technique

Algorithm 2 LSDE Multiplier

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

can also be used for the LSDE, like in the case of the LSE multiplier. We end this section by noticing that, as in Algorithm 1, if C is initialized to I in Algorithm 2, we can obtain as an output $A \cdot B + I \bmod p(\alpha)$ at no additional (hardware or delay) cost. This operation, known as a multiply/accumulate operation, is very useful in elliptic curve based systems.

Reduction mod $p(\alpha)$ for Digit Multipliers. In LSDE multipliers the product $W\alpha^D \bmod p(\alpha)$ occurs. As in the LSE multiplier case, one can derive equations for the modular reduction for *particular* irreducible $p(\alpha)$ polynomials. However, it is more interesting to search for polynomials that minimize the complexity of the reduction operation. In coming up with these optimum irreducible

polynomials we use two theorems from [33], adapted to the case of $GF(p^m)$ fields with p odd.

Theorem 1. *Assume that* $p(\alpha) = \alpha^m + p_k\alpha^k + \sum_{j=0}^{k-1} p_j\alpha^j$, *with* $k < m$. *For* $t \leq m-1-k$, *the degree of* α^{m+t} *can be reduced to be less than* m *in one step with the following equation:*

$$\alpha^{m+t} \bmod p(\alpha) = -p_k\alpha^{k+t} - (\sum_{j=0}^{k-1} p_j\alpha^{j+t}) \tag{7}$$

Theorem 2. *For Digit multipliers with digit-element size* D, *when* $D \leq m-k$ *the degree of the intermediate results in Algorithm 2 can be reduced to be less than* m *in one step.*

Theorems 1 and 2, implicitly say that for a given irreducible polynomial $p(\alpha) = \alpha^m + p_k\alpha^k + \sum_{j=0}^{k-1} p_j\alpha^j$, the digit-element size will depend on the value of k.

Area/Time Complexity of LSDE Multipliers. Before estimating the complexity of the LSDE multiplier, it is helpful to obtain equations to describe the values of A and C at iteration i in Algorithm 2. Thus, assume that B_i is as in (5), $p(\alpha)$ as in Theorem 1, $A = \sum_{i=0}^{m-1} a_i\alpha^i$, and $D \leq m-k$ (Theorem 2). Then,

$$C^{(i)} = D^{(i)} + C^{(i-1)} = \sum_{j=0}^{m+D-2} \left(d_j^{(i)} + c_j^{(i-1)}\right)\alpha^j \tag{8}$$

$$A^{(i)} = \sum_{j=D}^{m-1} a_{j-D}^{(i-1)}\alpha^j + \sum_{s=0}^{k}\sum_{j=0}^{D-1} \left(-p_s \cdot a_{j+m-D}^{(i-1)}\right)\alpha^{j+s} \tag{9}$$

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

$$D^{(i)} = \sum_{j=0}^{m+D-2} d_j^{(i)}\alpha^j = B_i \cdot A^{(i-1)} = \sum_{j=0}^{m-1}\sum_{k=0}^{D-1} \left(a_j^{(i-1)} \cdot b_{Di+k}\right)\alpha^{j+k} \tag{10}$$

Now it is easy to see that in each iteration one requires mD multipliers in parallel and $\sum_{j=0}^{D-2} j + \sum_{j=D-1}^{m-1}(D-1) + \sum_{j=m}^{m+D-2}(m+D-2-j) = (D-1)(m-1)$ adders. Therefore, according to (8), we only need $(D-1)(m-1) + m + D - 1 = mD$ adders to compute $C^{(i)}$. Using a ripple adder architecture, the critical path is given by $D-1$ adder delays from the computation of $D^{(i)}$, one adder delay from the computation $d_j^{(i)} + c_j^{(i-1)}$ in (8), and one multiplier. We notice, however, that it is possible to improve the critical path delay of the LSDE multiplier by using a binary tree of adders[4]. Using this technique one would reduce the length of the critical path from D $GF(p)$ adders and one $GF(p)$ multiplier to $\lceil \log_2(D+1) \rceil$ adders and one multiplier. We use this, as our complexity for the critical path.

[4] Binary trees have been used both in [33] and [26] in the context of $GF(2^n)$ arithmetic to reduce delay and power consumption.

Table 2. Area complexity and critical path delay of LSDE multiplier

Irreducible polynomial	Area Complexity	Critical Path Delay	Latency (# clocks)
r-nomial	$(m+r-2)D$ ADD + $(m+r-1)D$ MUL	$\lceil \log_2(D+1) \rceil$ ADD + 1 MUL	$\lceil \frac{m}{D} \rceil$
General	$(m+k)D$ ADD + $(m+k+1)D$ MUL	$\lceil \log_2(D+1) \rceil$ ADD + + 1 MUL	$\lceil \frac{m}{D} \rceil$

The computation of (9) requires only $D(k+1)$ multipliers (notice that the second term of (9) looks exactly the same as $D^{(i)}$ in (10), except that the limits in the summation are changed) and at most Dk adders. In the case in which $p(\alpha)$ is an r-nomial, the complexities reduce to $(r-1)D$ multipliers and at most $(r-2)D$ adders (notice that the first summation in (9) starts at $j = D$, thus, the first D coefficients resulting from the second summation do not need to be added to anything). We say *at most* because depending on the values of D and k, some adders may be saved. These results are summarized in Table 2

Table 2 makes the same assumptions as for the LSE case, i.e., ADD and MUL refer to the area and delay of a $GF(p)$ adder and multiplier, respectively, delay or area of storage elements are not taken into account, and no distinction is made between general and constant $GF(p)$ multipliers. We end by noticing that an LSDE multiplier with $D = 1$ is equivalent to an LSE multiplier. Our complexity estimates verify this if you let $D = 1$ and $k = m - 1$ in Table 2.

4.3 Comments on Irreducible Polynomials for $GF(p^m)$

From Theorems 1 and 2, it is obvious that choosing an irreducible polynomial should be carefully done. For fields $GF(p^m)$ with odd prime characteristic it is often possible to choose irreducible binomials $p(\alpha) = x^m - \omega$, $\omega \in GF(p)$. This is particularly interesting since binomials are never irreducible in characteristic 2 fields. Another interesting property of binomials is that they are optimum from the point of view of Theorem 1. In particular for any irreducible binomial $p(\alpha) = x^m - \omega$, $k = 0$ and $D \leq m$ in Theorem 2, which means that even in the degenerate case where $D = m$ (i.e. a parallel multiplier) one is able to perform the reduction in one step. In addition, reduction is virtually for free, corresponding to just a few $GF(p)$ multiplications (this follows from the fact that $\alpha^m = \omega$). A specific sub-class of these fields where q is a prime of the form $q = p = 2^n - c$, c "small", has recently been proposed for cryptographic applications in [4]. We notice that the existence of irreducible binomials has been completely established as Theorem 3 shows[5].

Theorem 3 ([22]). *Let $m \geq 2$ be an integer and $\omega \in F_q^\star$. Then the binomial $x^m - \omega$ is irreducible in $F_q[x]$ if and only if the following two conditions*

[5] Reference [22] is used here as a convenient reference for well established results.

are satisfied: (i) each prime factor of m divides the order e of ω in $F_q^\star$, but not $(q-1)/e$; (ii) $q \equiv 1 \bmod 4$ if $m \equiv 0 \bmod 4$.

When irreducible binomials can not be found, one searches in incremental order for irreducible trinomials, quadrinomials, etc. In [15] von zur Gathen and Nöcker conjecture that the minimal number of terms $\sigma_q(m)$ in irreducible polynomials of degree m in $GF(q)$, q a power of a prime, is for all $m \geq 1$, $\sigma_2(m) \leq 5$ and $\sigma_q(m) \leq 4$ for $q \geq 3$. This conjecture has been verified for $q = 2$ and $m \leq 10000$ [6, 11, 15, 36, 37, 38] and for $q = 3$ and $m \leq 539$ [14].

By choosing irreducible polynomials with the least number of non-zero coefficients, one can reduce the area complexity of the LSDE multiplier (this follows directly from Table 2). We point out that by choosing irreducible polynomials such that their non-zero coefficients are all equal to $p-1$ one can further reduce the complexity since all the multiplications by $-p_s$ in (9) reduce to multiplication by 1. Notice that there is no existence criteria for irreducibility of trinomials over any field $GF(p^m)$. The most recent advances in this area are the results of Loidreau [23], where a table that characterizes the parity of the number of factors in the factorization of a trinomial over $GF(3)$ is given, and the necessary (but not sufficient) irreducibility criteria for trinomials introduced by von zur Garten in [14]. Neither reference provides tables of irreducible polynomials.

5 Case Study: $GF(3^m)$ Arithmetic

FPGAs are reconfigurable hardware devices whose basic logic elements are Look-Up Tables (LUTs), also called Configurable Logic Blocks (CLBs), flip-flops (FFs), and, for modern devices, memory elements [1, 2, 35]. The LUTs are used to implement Boolean functions of their inputs, that is, functions traditionally implemented with logic gates. In the particular case of the XCV1000E-8-FG1156 and the XC2VP20-7-FF1156, their basic building blocks are 4-input bits/1-output bit LUTs. This means that all basic arithmetic operations in $GF(3)$ (add, subtract, and multiply) can be done with 2 LUTs, where each LUT generates one bit of the output.

5.1 Cubing in $GF(3^m)$

It is well known that for $A \in GF(p^m)$ the computation of A^p is linear. In the particular case of $p = 3$, we can write the frobenius map as:

$$A^3 \equiv \left(\sum_{i=0}^{m-1} a_i \alpha^i \right)^3 \bmod p(\alpha) = \sum_{i=0}^{m-1} a_i \alpha^{3i} \bmod p(\alpha) \tag{11}$$

Equation (11) can in turn be written as the sum of three terms (where we have re-written the indices in the summation):

$$A^3 \equiv \sum_{\substack{i=0 \\ i \equiv 0 \bmod 3}}^{3(m-1)} a_{\frac{i}{3}} \alpha^i \bmod p(\alpha) \equiv T + U + V \bmod p(\alpha) \tag{12}$$

$$\equiv \left(\sum_{\substack{i=0 \\ i\equiv 0 \bmod 3}}^{m-1} a_{\frac{i}{3}} \alpha^i \right) + \left(\sum_{\substack{i=m \\ i\equiv 0 \bmod 3}}^{2m-1} a_{\frac{i}{3}} \alpha^i \right) + \left(\sum_{\substack{i=2m \\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{\frac{i}{3}} \alpha^i \right) \bmod p(\alpha)$$

Notice that only U and V need to be reduce mod $p(\alpha)$. We further assume that $p(\alpha) = x^m + p_t x^t + p_0$ with $t < m/3$. This assumption is valid in terms of the existence of such irreducible trinomials as shown in Section 5.2. Thus,

$$U = \sum_{\substack{i=m \\ i\equiv 0 \bmod 3}}^{2m-1} a_{\frac{i}{3}} \alpha^i \bmod p(\alpha) = \sum_{\substack{i=m \\ i\equiv 0 \bmod 3}}^{2m-1} a_{\frac{i}{3}} \alpha^{i-m} \left(-p_t \alpha^t - p_0\right) \bmod p(\alpha)$$

$$V = \sum_{\substack{i=2m \\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{\frac{i}{3}} \alpha^i \bmod p(\alpha) = \sum_{\substack{i=2m \\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{\frac{i}{3}} \alpha^{i-2m} \left(\alpha^{2t} - p_t p_0 \alpha^t + 1\right) \bmod p(\alpha)$$

where we have made use of the fact that $(-p_t\alpha^t - p_0)^2 = (\alpha^{2t} - p_t p_0 \alpha^t + 1)$ in $GF(3)$. It can be shown that U and V can be reduced to be of degree less than m in one extra reduction step. To estimate the complexity of this cubing circuit, we assume that $p(\alpha)$ is a fixed irreducible trinomial with $t < m/3$ i.e., that multiplications in $GF(3)$ (for example multiplying by $-p_t p_0$) can be handled by adders and subtracters. Then, it can be shown that one needs in the order of $2m$ adders/subtracters to perform a cubic operation in $GF(3^m)$.

5.2 Irreducible Polynomials over $GF(3)$

Following the criteria of Section 4.3 for choosing irreducible polynomials, we tried to find irreducible binomials first. Unfortunately, the only irreducible binomial over $GF(3)$ is x^2+1, thus we considered irreducible trinomials. Notice that $x^m + x^t + 1$ is never irreducible over $GF(3)$ since 1 is always a root of it. Thus, we only searched for irreducible trinomials of the following forms: $x^m - x^t - 1$ or $x^m \pm x^t \mp 1$. For $2 \leq m \leq 255$, we exhaustively searched for these trinomials (see Tables 6, 7, and 8 in Appendix A). There are only 23 degrees m in the range above for which we were unable to find trinomials (which agrees with the findings in [14]) and thus quadrinomials can be found. Of these quadrinomials only 4 correspond to m prime $(149, 197, 223, 233)$. Prime m is the most commonly used degree in cryptographic applications. We notice that of the 50 primes in the above range which had trinomials, we were not able to find trinomials with $t < m/3$ for 9 of them (18 %).

6 $GF(3^m)$ Prototype Implementation and Comparisons

Figure 1 shows a block diagram of the prototyped arithmetic unit (AU). Notice that in Figure 1, all bus-widths correspond to how many $GF(3)$ elements can be carried by the bus, i.e., if we write m, then it is understood that the bus is $2m$ bits wide. The AU consists of an LSDE multiplier and a cubing circuit.

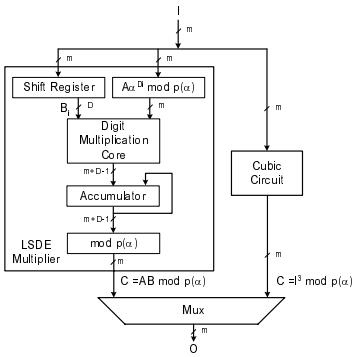

Fig. 1. $GF(3^m)$ arithmetic unit architecture

The multiplier and the cubing circuit support the computation of field additions, squares, multiplications, and inversions. For addition and subtraction we take advantage of the multiply/accumulate capabilities of the LSDE multiplier and cubing circuit. In other words, the addition $C = A+B$ is done by first computing $A{\cdot}1$ and then adding to it the product $B{\cdot}1$. This takes two clock cycles. However, if operand A is already in the accumulator of the multiplier one can compute $C = B \cdot 1 + A$ in one clock. This eliminates the need for an adder. Subtractions are computed in a similar manner, i.e., $C = A-B$ is done by first computing $A{\cdot}1$ and then adding to it the product $(-1) \cdot B$ or alternatively as $C = (-1) \cdot B + A$.

AU prototypes were developed to verify the suitability of the architecture shown in Figure 1 for reconfigurable FPGA logic and compare the efficiency of $GF(3^m)$ and $GF(2^m)$ AUs. The prototypes were coded in VHDL at a very low level. The VHDL code was synthesized using Synopsis FPGA Compiler 3.7.1 and the component placement and routing was done using Xilinx Design Manager 4.2.03i. The prototypes were synthesized and routed for the Xilinx XCV1000-8-FG1156 and XC2VP20-7-FF1156 FPGAs. The XCV1000E-8-FG1156 prototype allowed us to compare our AU implementations against the AU for $GF(2^m)$ used in the EC processor (ECP) from [27], which is one of the fastest ECP implemented in FPGA logic for EC defined over fields $GF(2^{167})$. The XC2VP20-7-FF1156 prototype allowed us to verify the speed of our AU for one of the newest families of Xilinx FPGAs. Three implementation were developed which support the fields $GF(3^{97})$, $GF(2^{151})$, and $GF(2^{241})$. The fields $GF(3^{97})$ and $GF(2^{241})$ are used in Weil and Tate pairing schemes for systems with comparable degrees of security (see [10, 5, 29]). The field $GF(2^{151})$ offers security comparable to that of $GF(3^{97})$ for cryptosystems based on the EC discrete logarithm problem.

6.1 $GF(3^m)$ Complexity Estimates

Table 3 shows the complexity estimates for the AU shown in Figure 1. The estimates assume the use of optimum irreducible polynomials and give the register complexity in terms of the number of flip-flops. Note that the register estimates

Table 3. $GF(3^m)$ AU complexity estimates

Circuit	Area Complexity
LSE Mult.	$(4mD + 2m + 6D)$ LUT + $6m + 2D + 4$ FF
Cubic	$4m$ LUT
Mux	$2m$ LUT
AU (total)	$(4mD + 8m + 6D)$ LUT + $6m + 2D + 4$ FF

do not account for registers used to reduce the critical path delay of a multiplier, a technique known as pipelining. This technique was used to reduce the critical path delay of the prototype implementations. The complexity estimates are based on the following assumptions:

1. $GF(3)$ adders, subtracters, or multipliers require two LUTs, including adders that add weighted inputs, for example, adders that compute $(a_i * c) + (b_i * d)$ where c and d are fixed constants. Also a 2:1 multiplexer requires one LUT.
2. From Table 2 the digit multiplication core and accumulator circuits require mD $GF(3)$ multipliers and mD $GF(3)$ adders. This circuit stores the result in two $(m+D-1)$-bit registers. An m-bit register requires m flip-flops (FFs).
3. The estimates for the $A\alpha^{Di} \bmod p(\alpha)$ circuit assume that the circuit contains two m-bit multiplexers that select between the element A and the element $A\alpha^{Di} \bmod p(\alpha)$. An m-bit multiplexer requires m LUTs. For programmable optimum irreducible trinomials, the circuitry that generates $A\alpha^{Di} \bmod p(\alpha)$ requires $2D$ $GF(3)$ multipliers and D adders (see Table 2). This circuit stores the result in two m-bit registers and the coefficients of $p(x)$ in two $2r$-bit registers ($r = 3$ for trinomials).
4. The coefficients of B are fed in by two m-bit parallel in/serial out shift registers. Each shift registers contains m 2:1 multiplexers and m registers.
5. The cubic circuit requires $2m$ $GF(3)$ adders.
6. The complexity for the $GF(2^m)$ AU is done according to [26]. It is also assumed that the $GF(2^m)$ AU contains an LSD multiplier and a squarer.

Table 4. AU estimated vs. measured complexity for prototypes ($D = 16$)

Circuit	Estimated complexity	Measured complexity (incl. pipelining registers & I/O)	LUT Estimate Error (measured/est.)
$GF(2^{151})$	2366 LUT + 453 FF ($15.7m$ LUT + $3m$ FF)	2239 LUT + 1092 FF ($14.3m$ LUT + $7.2m$ FF)	-5.4 %
$GF(2^{241})$	3705 LUT + 723 FF ($14.9m$ LUT + $3m$ FF)	3591 LUT + 1722 FF ($15.4m$ LUT + $7.1m$ FF)	-3.1 %
$GF(3^{97})$	7080 LUT + 618 FF ($73.0m$ LUT + $6.4m$ FF)	7122 LUT + 2790 FF ($73.4m$ LUT + $7.2m$ FF)	1.0 %

The estimates that we obtain from our models are very accurate when compared to the actual measured complexities. This validates our models and assumptions.

6.2 Results

Table 5 presents the timings obtained for our three prototypes. We have tried to implement our designs in such a way that we can make a meaningful comparison. Thus, although, the clock rates are not exactly the same between the different designs (this is due to the fact that the clock rate depends on the critical path of the AU which is different for each circuit), they are not more than 10 % different. The platforms are the same and we chose same digit sizes for both $GF(2^m)$ and $GF(3^m)$ architectures. The results make sense, for the same digit size ($D = 16$) we obtain that the $GF(3^{97})$ design is about twice as big as the $GF(2^{241})$ design and more than 3 times the size of the $GF(2^{151})$ AU. This is offset by the gain in performance. At similar clock rates the $GF(3^{97})$ design is 2.7 times faster than the corresponding $GF(2^{241})$ AU and 1.4 times faster than the $GF(2^{151})$ one. It is clear that by using more hardware resources for $GF(3^{97})$ we achieve better performance than characteristic two fields. In particular, by choosing the same digit size for both fields, we implicitly process twice as many bits of the multiplier in $GF(3^{97})$ as in the $GF(2^m)$ case (remember that the E in LSDE refers to elements of $GF(p)$ and not bits as in the $GF(2^m)$ case). Table 5 also includes the results from [29]. We don't think it is possible to make a meaningful comparison, other than point out that by coding directly in VHDL, one can achieve huge improvements in performance of FPGA based implementations.

7 Conclusions

In this paper, we have generalized the finite field multipliers of [33] to the odd characteristic case. We have also presented multiplication algorithms for both serial and digit-based architectures. Finally, we have presented a general framework to choose irreducible polynomials that reduce the computation of the modulo reduction operation. More importantly, we have shown that it is possible to achieve considerable performance from FPGA implementations of non-binary finite fields. However, from our discussion in the previous section, we conclude that fields of characteristic 2 can not be surpassed by other fields if one considers both area and time performance measures.

Table 5. Comparison of multiplication time for $GF(2^{151})$,$GF(2^{241})$, and $GF(3^{97})$ prototypes ($D = 16$) and the AU from [29]

Circuit	Time for optimized mult. [29] [29](in usecs)	Mult. time for prototypes	
		XCV1000-8-FG1156 (in usecs)	XC2VP20-7-FF1156 (in usecs)
$GF(2^{151})$	N/A	0.139 (@ 71.7 MHz)	0.100 (@ 100.2 MHz)
$GF(2^{241})$	37.32 (@ 20 MHz)	0.261 (@ 61.3 MHz)	0.150 (@ 107 MHz)
$GF(3^{97})$	50.68 (@ 20 MHz)	0.097 (@ 72 MHz)	0.074 (@ 94.4 MHz)

References

[1] Actel Corporation. *Actel's ProASIC Family, The Only ASIC Design Flow FPGA*, 2001. 164, 168

[2] Altera Corporation. *APEX 20KC Programmable Logic Device Data Sheet*, 2001. 164, 168

[3] D. V. Bailey and C. Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume LNCS 1462, pages 472–485, Berlin, Germany, 1998. Springer-Verlag. 159

[4] 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. 159, 167

[5] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume LNCS 2442, pages 354–368. Springer-Verlag, 2002. 159, 170

[6] Blake, Gao, and Lambert. Constructive problems for irreducible polynomials over finite fields. In *Information Theory and Applications*, pages 1–23. Springer-Verlag, 1993. 168

[7] D. Boneh, B.Lynn, and H. Shacham. Short signatures from the Weil pairing. In J. Boyd, editor, *Advances in Cryptology — Asiacrypt 2001*, volume LNCS 2148, pages 514–532. Springer-Verlag, 2001. 159

[8] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume LNCS 2139, pages 213–229. Springer-Verlag, 2001. 159

[9] E. D. Di Claudio, F. Piazza, and G. Orlandi. Fast Combinatorial RNS Processors for DSP Applications. *IEEE Transactions on Computers*, 44(5):624–633, May 1995. 159, 164

[10] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate Pairing. In C. Fieker and D. Kohel, editors, *Algorithmic Number Theory — ANTS-V*, volume LNCS 2369, pages 324–337. Springer-Verlag, 2002. 159, 160, 170

[11] S. W. Golomb. *Shift Register Sequences*. Holden-Day, San Francisco, USA, 1967. 168

[12] J. Guajardo and C. Paar. Itoh-Tsujii Inversion in Standard Basis and Its Application in Cryptography and Codes. *Design, Codes, and Cryptography*, 25(2):207–216, February 2002. 162

[13] J. Guajardo, T. Wollinger, and C. Paar. Area Efficient $GF(p)$ Architectures for $GF(p^m)$ Multipliers. In *Proceedings of the 45th IEEE International Midwest Symposium on Circuits and Systems — MWSCAS 2002*, August 2002. 159, 164

[14] J. von zur Gathen. Irreducible Trinomials over Finite Fields. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation — ISSAC2001*, pages 332–336. ACM Press, 2001. 168, 169

[15] J. von zur Gathen and M. Nöcker. Exponentiation in Finite Fields: Theory and Practice. In T. Mora and H. Mattson, editors, *Applied Algebra, Agebraic Algorithms and Error Correcting Codes — AAECC-12*, volume LNCS 1255, pages 88–113. Springer-Verlag, 2000. 168

[16] S. K. Jain and K. K. Parhi. Efficient standard basis reed-solomon encoder. In *1996 IEEE International Conference of Acoustics, Speech, and Signal Processing*, Atlanta, May 1996. 163

[17] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory — ANTS-IV*, volume LNCS 1838, pages 385–394. Springer-Verlag, 2000. 159
[18] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Sov. Phys. Dokl. (English translation)*, 7(7):595–596, 1963. 160
[19] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987. 158
[20] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):129–150, 1989. 158
[21] N. Koblitz. An elliptic curve implementation of the finite field digital signature algorithm. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO 98*, volume LNCS 1462, pages 327–337. Springer-Verlag, 1998. 159
[22] R. Lidl and H. Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, Massachusetts, USA, 1983. 161, 167
[23] P. Loidreau. On the Factorization of Trinomials over F_3. Rapport de recherche no. 3918, INRIA, April 2000. 168
[24] V. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, volume LNCS 218, pages 417–426, Berlin, Germany, 1986. Springer-Verlag. 158
[25] Jin Young Oo, Young-Gern Kim, Dong-Young Park, and Heung-Su Kim. Efficient Multiplier Architecture Using Optimized Irreducible Polynomial over $GF((3^n)^3)$. In *Proceedings of the IEEE Region 10 Conference – TENCON 99. "Multimedia Technology for Asia-Pacific Information Infrastructure"*, volume 1, pages 383–386, 1999. 160
[26] G. Orlando. *Efficient Elliptic Curve Processor Architectures for Field Programmable Logic*. PhD thesis, Dept. of ECE, Worcester Polytechnic Institute, March 2002. 166, 171
[27] 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 LNCS 1965, pages 41–56. Springer-Verlag, 2000. 170
[28] P. Mihăilescu. Optimal Galois Field Bases which are not Normal. Recent Results Session — FSE '97, 1997. 159
[29] 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, *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume LNCS. Springer-Verlag, 2002. 159, 160, 161, 170, 172
[30] V. Paliouras, K. Karagianni, and T. Stouraitis. A Low-Complexity Combinatorial RNS Multiplier. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(7):675–683, July 2001. 159, 164
[31] M. G. Parker and M. Benaissa. $GF(p^m)$ Multiplication Using Polynomial Residue Number Systems. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 42(11):718–721, November 1995. 160
[32] N. Smart. Elliptic Curve Cryptosystems over Small Fields of Odd Characteristic. *Journal of Cryptology*, 12(2):141–151, Spring 1999. 159
[33] 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. 159, 161, 162, 164, 165, 166, 172
[34] E. Verheul. Self-blindable Credential Certificates from the Weil Pairing. In C. Boyd, editor, *Advances in Cryptology — Asiacrypt 2001*, volume LNCS 2248, pages 533–551. Springer-Verlag, 2001. 159

[35] Xilinx, Inc. *The Programmable Logic Data Book*, 2000. 164, 168
[36] N. Zierler. On $x^n + x + 1$ over $GF(2)$. *Information and Control*, 16:67–69, 1970. 168
[37] N. Zierler and J. Brillhart. On Primitive Trinomials (mod2). *Information and Control*, 13:541–554, 1968. 168
[38] N. Zierler and J. Brillhart. On Primitive Trinomials (mod 2), II. *Information and Control*, 14:566–569, 1969. 168

A Lists of Irreducible Polynomials over $GF(3)$

Table 6. Irreducible trinomials of the form $x^m + x^t + 2$ over $GF(3)$, $2 \leq m \leq 255$

m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t
2	1	22	5	43	26	71	20	93	70	117	52	143	108	167	92	187	8	211	122	238	5
3	2	23	8	44	3	72	28	95	48	119	2	144	56	168	28	188	11	214	65	239	24
4	1	24	4	45	28	73	30	96	16	120	4	145	24	169	24	191	116	215	36	240	8
5	4	25	6	46	5	76	9	97	12	121	40	147	8	170	43	192	32	216	4	241	88
6	1	26	7	47	32	77	16	99	74	124	25	148	3	171	20	193	12	217	132	242	115
7	2	27	20	48	8	78	13	100	25	125	52	150	73	172	19	194	55	219	26	243	122
8	2	28	2	51	50	79	26	101	70	126	49	151	2	173	166	195	26	220	15	244	31
9	4	29	4	52	7	80	2	102	25	127	8	152	18	174	73	196	79	222	89	245	148
11	2	30	1	53	22	81	40	103	50	128	6	153	94	176	12	198	29	224	12	246	13
12	2	31	20	54	1	83	32	104	5	131	48	155	12	177	52	199	164	225	16	247	122
13	4	32	5	55	26	84	14	107	32	133	88	156	26	178	11	200	3	227	68	248	50
14	1	33	28	56	3	85	16	108	2	134	61	157	22	179	104	201	88	228	14	249	76
15	2	35	2	59	20	86	13	109	88	135	44	158	61	180	38	203	8	229	72	251	26
16	4	36	14	60	2	87	26	111	2	136	57	159	32	181	40	204	50	230	73	252	98
17	16	37	6	61	30	88	6	112	6	137	136	160	4	182	25	205	78	232	30	253	12
18	7	39	26	63	26	89	64	113	70	139	80	162	19	183	2	206	61	234	91	254	73
19	2	40	1	64	3	90	19	114	7	140	59	163	80	184	20	208	10	235	26	255	26
20	5	41	40	67	2	91	74	115	32	141	64	164	15	185	64	209	40	236	9		
21	16	42	7	69	52	92	10	116	15	142	65	165	22	186	47	210	7	237	70		

Table 7. Irreducible trinomials of the form $x^m + 2x^t + 1$ over $GF(3)$, $3 \leq m \leq 255$

m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t
3	1	25	3	46	6	71	20	94	30	119	2	145	24	170	32	194	24	222	4	249	59
5	1	26	2	47	15	73	1	95	47	121	1	146	2	171	20	195	26	225	16	250	104
6	2	27	7	50	6	74	12	97	12	122	14	147	8	173	7	198	38	226	38	251	9
7	2	29	4	51	1	77	16	99	19	125	52	151	2	174	52	199	35	227	11	253	7
9	4	30	4	53	13	78	14	101	31	126	52	153	59	177	52	201	88	229	72	254	16
10	2	31	5	54	14	79	26	102	2	127	8	154	32	178	26	202	62	230	64	255	26
11	2	33	5	55	11	81	40	103	47	131	27	155	12	179	59	203	3	234	104		
13	1	34	2	58	8	82	2	106	26	133	15	157	22	181	37	205	9	235	26		
14	4	35	2	59	17	83	27	107	3	134	4	158	52	182	34	206	94	237	70		
15	2	37	6	61	7	85	16	109	9	135	44	159	32	183	2	209	40	238	4		
17	1	38	4	62	10	86	34	110	22	137	1	162	80	185	64	211	89	239	5		
18	8	39	7	63	26	87	26	111	2	138	34	163	59	186	46	214	6	241	88		
19	2	41	1	66	10	89	13	113	19	139	59	165	22	187	8	215	36	242	2		
21	5	42	10	67	2	90	34	115	32	141	5	166	54	190	94	217	85	243	121		
22	4	43	17	69	17	91	17	117	52	142	40	167	71	191	71	218	18	245	97		
23	3	45	17	70	4	93	23	118	34	143	35	169	24	193	12	219	25	247	122		

Table 8. Irreducible trinomials of the form $x^m + 2x^t + 2$ over $GF(3)$, $2 \leq m \leq 255$

m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t
2	1	22	5	43	17	71	51	93	23	117	65	143	35	167	71	187	65	211	89	238	5
3	1	23	3	44	3	72	28	95	47	119	3	144	56	168	28	188	11	214	65	239	5
4	1	24	4	45	17	73	1	96	16	120	4	145	73	169	37	191	71	215	59	240	8
5	1	25	3	46	5	76	9	97	81	121	1	147	43	170	43	192	32	216	4	241	117
6	1	26	7	47	15	77	25	99	19	124	25	148	3	171	151	193	81	217	85	242	115
7	5	27	7	48	8	78	13	100	25	125	73	150	73	172	19	194	55	219	25	243	121
8	2	28	2	51	1	79	53	101	31	126	49	151	125	173	7	195	49	220	15	244	31
9	5	29	25	52	7	80	2	102	25	127	119	152	18	174	73	196	79	222	89	245	97
11	3	30	1	53	13	81	41	103	47	128	6	153	59	176	12	198	29	224	12	246	13
12	2	31	5	54	1	83	27	104	5	131	27	155	129	177	83	199	35	225	209	247	125
13	1	32	5	55	11	84	14	107	3	133	15	156	26	178	11	200	3	227	11	248	50
14	1	33	5	56	3	85	31	108	2	134	61	157	69	179	59	201	113	228	14	249	59
15	7	35	17	59	17	86	13	109	9	135	91	158	61	180	38	203	3	229	79	251	9
16	4	36	14	60	2	87	37	111	13	136	57	159	127	181	37	204	50	230	73	252	98
17	1	37	13	61	7	88	6	112	6	137	1	160	4	182	25	205	9	232	30	253	7
18	7	39	7	63	37	89	13	113	19	139	59	162	19	183	181	206	61	234	91	254	73
19	11	40	1	64	3	90	19	114	7	140	59	163	59	184	20	208	10	235	83	255	229
20	5	41	1	67	11	91	17	115	83	141	5	164	15	185	121	209	49	236	9		
21	5	42	7	69	17	92	10	116	15	142	65	165	77	186	47	210	7	237	167		