Efficient Multiplier Architectures for Galois Fields $GF(2^{4n})$

Christof Paar, *Member*, *IEEE*, Peter Fleischmann, and Peter Roelse

Abstract—This contribution introduces a new class of multipliers for finite fields $GF((2^n)^4)$. The architecture is based on a modified version of the Karatsuba-Ofman algorithm (KOA). By determining optimized field polynomials of degree four, the last stage of the KOA and the modulo reduction can be combined. This saves computation and area in VLSI implementations. The new algorithm leads to architectures which show a considerably improved gate complexity compared to traditional approaches and reduced delay if compared with KOA-based architectures with separate modulo reduction. The new multipliers lead to highly modular architectures and are, thus, well suited for VLSI implementations. Three types of field polynomials are introduced and conditions for their existence are established. For the small fields, where $n = 2, 3, \ldots, 8$, which are of primary technical interest, optimized field polynomials were determined by an exhaustive search. For each field order, exact space and time complexities are provided.

Index Terms—Galois fields, composite fields, multiplication, Karatsuba Ofman, modulo reduction, bit parallel, VLSI architecture.

1 Introduction

Galois fields have gained wide spread applications in modern communication systems, such as computer networks, satellite links, or compact disks. They all use finite field arithmetics for error correction [32] or for cryptographic algorithms [22]. Both issues are central for the provision of reliable and secure communication. Further applications can be found in signal processing [3] and pseudorandom number generation [30]. Modern applications in many cases call for VLSI implementations of the arithmetic modules in order to satisfy the high speed requirements. For technical applications, extension fields of $GF(2)$, denoted by $GF(2^k)$, are of primary interest. Assuming a basis representation of the field elements, addition is a relatively inexpensive operation, whereas the other field operation, multiplication, is costly in terms of gate count and delay. Efficient multiplier architectures are especially crucial from a system design point of view, since most advanced arithmetic functions, such as inversion and exponentiation, are based on multiplication. There have been considerable research efforts in the development of VLSI-suited multiplier architectures over the last decade, which is reflected by numerous journal publications [31], [29], [11], [12], [20], [10], [9], [5] and recent dissertations [21], [8], [6], [24], [13]. Multipliers can be classified into bit serial and bit parallel architectures. Since there is a space-time trade-off, bit parallel multipliers tend to be faster, which makes them attractive for many applications. The penalty, however, is higher hardware costs. This article will focus on bit parallel architectures with an optimized gate count.

- *C. Paar is with the Electrical and Computer Engineering Department, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609. E-mail: christof@ece.wpi.edu.*
- *P. Fleischmann and P. Roelse are with the Institute for Experimental Mathematics, University of Essen, Ellernstr. 29, 45326 Essen, Germany.*

Manuscript received 13 Mar. 1996; revised 13 Jan. 1997.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 102018.

There are three different popular basis representations for elements of $GF(2^k)$, canonical (or standard), dual, and normal basis. A field element in either basis is represented by k bits. Most bit parallel multiplier architectures show a computational complexity of $O(k^2)$ operations in $GF(2)$; canonical and dual basis architectures are lower bounded by k^2 multiplications and $k^2 - 1$ additions, normal basis ones by k^2 multiplications and $2k^2 - k$ additions. A multiplication in $GF(2)$ can be realized by a two-input AND gate and an adder by a two-input XOR gate. In [27], it is shown that the computational complexity is also a good estimate for the chip area needed in VLSI implementations. For this reason, it is attractive to provide architectures with low computational complexity for efficient hardware implementations.

More recently, there have also been proposals for multipliers with complexities below the k^2 bound. These architectures are either based on multiple field extensions [2], [28], on fast convolution algorithms [1], or on both [25] (see [24, Section 3.2] for an overview.) One algorithm which was found to be particularly suited in this context is the Karatsuba-Ofman algorithm (KOA), which will be looked at in more detail in the following section.

In this contribution, a new architecture will be developed which further improves the computational complexity of the multipliers proposed in [1], [25], which use the standard KOA. The improvement is based on special classes of polynomials of degree four over fields $GF(2^n)$. These polynomials combine the structure of the KOA and the fact that the field characteristic is two. Fields of the type $GF(2^n)^4)$ are of great technical interest, since all fields whose elements are represented by multiples of bytes (i.e., 8, 16, 24, … bits) are of this form.

The outline of the remaining paper is the following. In Section 2, the multiplier from [25] will be reviewed for the case where the field has the form $GF(2^n)^4)$. Section 3 develops the underlying idea of the new architecture and shows how suitable irreducible polynomials over $GF(2^n)$ were determined.

In Section 4, different classes of efficient field polynomials are introduced together with proofs of existence. Expressions for the corresponding multiplier architectures are derived. In Section 5, overall gate and time complexity of multipliers in $GF(2^k)$, $k = 8, 12, \ldots, 32$ are provided.

2 Multiplication in Composite Fields $GF(2^n)^4)$

This section will review the multiplier proposed in [25]. The basic idea is to represent the field $GF(2^k)$ by a so-called composite field [7] of the form $GF(2^n)^m)$ and to apply a fast convolution algorithm to field elements in polynomial (or canonical) representation.

The field $GF(2^n)^4)$ is isomorphic to $GF(2^n)/(P(x))$, where $P(x)$ is an irreducible polynomial of degree four over $GF(2^n)$. In the following, a residue class will be identified with the polynomial of least degree in this class. An element of $GF(2^n)^4)$ can then be represented by a polynomial with a maximum degree of three with coefficients in $GF(2^n)$:

$$A(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \ ; \ a_i \in GF\left(2^n\right) \ ; \ A \in GF\left(\left(2^n\right)^4\right).$$

Field multiplication $C(x) = A(x)B(x)$, where $A, B, C \in GF(2^n)^m)$, can be performed by

$$C(x) = A(x) \times B(x) \bmod P(x). \tag{1}$$

The two steps involved in operation (1) are

1) Ordinary polynomial multiplication ($\times$),
2) Reduction modulo the field polynomial (mod).

Both operations require arithmetic with the polynomial coefficients which are elements of the subfield $GF(2^n)$. Hence, multiplication of coefficients $a_i \times b_j$ requires at least n^2 elementary multiplications (AND) and $n^2 - 1$ elementary additions (XOR), whereas addition of coefficients $a_i + b_j$ can be realized with only n elementary additions (XOR.)

In this situation—where coefficient multiplication is more costly than addition—the Karatsuba-Ofman algorithm (KOA) [14], [15, p. 278] can successfully be applied to the first steps of (1), the polynomial multiplication. The algorithm performs polynomial multiplication with a reduced number of coefficient multiplication at the cost of an increased number of coefficient additions. For the case considered here, i.e., polynomials of degree three, the KOA can be performed with $\log_2 4 = 2$ iterations. The complexity of the algorithm is given by the following lemmas.

LEMMA 1. *Two arbitrary polynomials in one variable of degree less or equal three with coefficients in $GF(2^n)$ can be multiplied by means of the Karatsuba-Ofman algorithm with:*

$$\#\otimes = 9, \tag{2}$$

$$\#\oplus \le 24, \tag{3}$$

multiplications and additions, respectively, in $GF(2^n)$.

LEMMA 2. *A parallel realization of the Karatsuba-Ofman algorithm for the multiplication of two arbitrary polynomials in one variable of degree less or equal three with coefficients in $GF(2^n)$ can be implemented with a time complexity (or delay) of:*

$$T = T_\otimes + 6T_\oplus, \tag{4}$$

where "$T_\otimes$" and "$T_\oplus$" denote the delay of one multiplier and one adder, respectively, in $GF(2^n)$.

PROOF. The lemmas follow immediately from Theorem 1 and 2, respectively, in [25] if $m = 4$. □

It should be noted that the number of additions in (3) can be reduced to 22 [25].

For the following sections, it will be beneficial to look at the exact operations which are performed by the KOA. The KOA operates in three stages. The first two stages split the two input polynomials $A(x)$ and $B(x)$ and perform multiplications. The result is a set of nine intermediate variables d_i, $i = 0, \ldots, 8$.

$$\begin{aligned}
d_0 &= a_0b_0 \\
d_1 &= (a_0 + a_1)(b_0 + b_1) \\
d_2 &= a_1b_1 \\
d_3 &= (a_0 + a_2)(b_0 + b_2) \\
d_4 &= (a_0 + a_1 + a_2 + a_3)(b_0 + b_1 + b_2 + b_3) \\
d_5 &= (a_1 + a_3)(b_1 + b_3) \\
d_6 &= a_2b_2 \\
d_7 &= (a_2 + a_3)(b_2 + b_3) \\
d_8 &= a_3b_3.
\end{aligned} \tag{5}$$

These equations can be realized with 10 additions and nine multiplications. A block diagram of a parallel realization of the two first two stages of the KOA is given in Fig. 1.

In the algorithm's third stage, the product polynomial is constructed. If the result of the multiplication is a product polynomial $C'(x)$ of degree six:

$$C'(x) = \sum_{i=0}^{6} c'_i x^i = A(x) \times B(x),$$

its coefficients can be expressed in terms of the d_i as:

$$\begin{aligned}
c'_0 &= d_0 \\
c'_1 &= d_0 + s_1 + d_2 \\
c'_2 &= d_0 + d_2 + d_3 + d_6 \\
c'_3 &= d_0 + d_1 + d_2 + d_3 + d_4 + d_5 + d_6 + d_7 + d_8 \\
c'_4 &= d_2 + d_5 + d_6 + d_8 \\
c'_5 &= d_6 + d_7 + d_8 \\
c'_6 &= d_8.
\end{aligned} \tag{6}$$

The equations can be realized with 12 additions if certain terms are precomputed.

In order to perform finite field multiplication, the polynomial $C'(x)$ has to be reduced modulo the field polynomial: $C(x) \equiv C'(x) \bmod P(x)$, where $C(x) = c_3x^3 + c_2x^2 + c_1x + c_0$. The modulo reduction can be viewed as a linear mapping of the seven coefficients c'_k to the four coefficients c_i of the form

$$c_i = c'_i + \sum_{j=0}^{2} r_{i,j}c'_{j+4} \ ; \ i = 0, \ldots, 3 \ ; \ r_{i,j} \in GF\left(2^n\right). \tag{7}$$

The reduction coefficients $r_{i,j}$ are functions of the field polynomial $P(x) = x^4 + \sum_{i=0}^{3} p_i x^i$, $p_i \in GF(2^n)$:

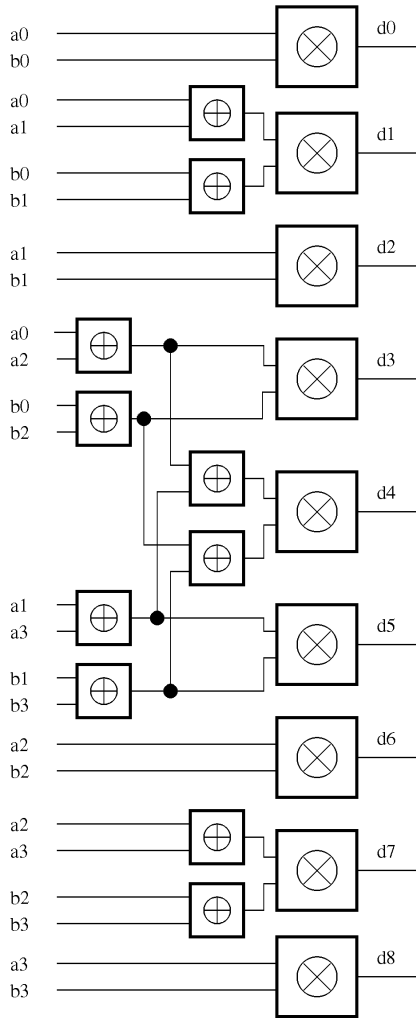

Fig. 1. Block diagram of the first two stages of the KOA for polynomials of degree three.

$$r_{i,j} = \begin{cases} p_i & ; \quad i = 0, \ldots, 3 \quad ; \quad j = 0; \\ r_{i-1,j-1} + r_{3,j-1} r_{i,0} & ; \quad i = 0, \ldots, 3 \quad ; \quad j = 1, 2; \end{cases} \tag{8}$$

where $r_{i-1,j-1} = 0$ if $i = 0$.

In [25, Table 1], suboptimum polynomials are listed which perform the mapping (7) with low complexity. They were determined by an exhaustive search. The overall computational complexity of a multiplier is the sum of the complexities from Lemma 1 and the complexity of a realization of (7).

3 A New Approach

In this section, a new approach is outlined. The new method combines properties of the KOA over fields of characteristic two and of the modulo reduction process.

In the architecture described above, polynomials were determined which optimize the complexity of the modulo reduction (7), i.e., the mapping from the c'_k to the c_i coefficients. This optimization is independent of the preceding polynomial multiplication by means of the KOA. The key idea of the new approach is to determine optimized mappings from the coefficients d_i, $i = 1, \ldots, 8$, to the coefficients c_i, $i = 0, \ldots, 3$, of the product polynomial. Such a mapping has the form

$$c_i = \sum_{j=0}^{8} s_{i,j} d_j \ ; \ i = 0, \ldots, 3 \ ; \ s_{i,j} \in GF\left(2^n\right). \tag{9}$$

The mapping includes the last stage of the KOA given in (6) and the modulo reduction as described by (7). Expressions for the coefficients $s_{i,j}$ are obtained by inserting (6) into (7). The potential advantage inherent in this approach is based on the following observation: If at least two coefficients $r_{i,j}$ and $r_{i,k}$ are equal, summations of the form $d_k + d_k$ may occur. Since the characteristic of the field $GF(2^n)$ is two, the equation $d_i + d_i = 0$ holds. Thus, cancellations can potentially occur in (9), leading to a lower computational complexity.

EXAMPLE. Let's assume there is a field polynomial $P(x)$ leading to coefficients $r_{3,0} = 1$ and $r_{3,1} = r_{3,2} = 0$ in (7). Then, the output coefficient c_3 can be expressed as

$$\begin{aligned} c_3 &= c'_3 + \sum_{j=0}^{2} r_{3,j} c'_{j+4} = c'_3 + c'_4 \\ &= \left(d_0 + d_1 + d_2 + d_3 + d_4 + d_5 + d_6 + d_7 + d_8\right) + \left(d_2 + d_5 + d_6 + d_8\right) \\ &= d_0 + d_1 + d_3 + d_4 + d_7, \end{aligned}$$

which can be realized with only four additions. In contrast, a straightforward realization of the sum $c'_3 + c'_4$ calls for 12 additions.

In order to find (9) with an optimized number of arithmetic operations, two different strategies were applied:

1) An *exhaustive search* through all irreducible polynomials of degree four over $GF(2^n)$, for $n \le 8$ was performed. For every polynomial the number of additions and constant multiplications required for (9) was evaluated. Besides the cancellation described above, further computational redundancies in the expressions were taken into account as follows: If there were identical $s_{i,j}$ for a given c_i, i.e., $s_{i,u} = s_{i,v}$, the number of constant multiplication is further reduced, since

$$c_i = \cdots + s_{i,u} d_u + \cdots + s_{i,v} d_v + \cdots = \cdots + s_{i,u}\left(d_u + d_v\right) + \cdots.$$

It was found that, for different values of n, similar polynomials lead to optimum architectures. By identifying these polynomial types, we were able to define classes of multiplier architectures which are very efficient computationally (type A and C in the following sections.)

2) Based on the results from the exhaustive search, we *constructed* another polynomial type (type B in the following) for a certain class of fields.

4 Efficient Multipliers

This section describes different optimized field polynomials and new classes of multipliers based on them.

4.1 Low Complexity Field Polynomials

This subsection introduces three types of irreducible polynomials and establishes their existence.

We begin with the definition of field polynomials for $GF(2^n)^4$).

DEFINITION 1. *Polynomials over $GF(2^n)$ of degree four of the form*

$$P_A(x) = x^4 + x^3 + x^2 + x + 1 \quad (10)$$

$$P_B(x) = x^4 + x^3 + (p+1)x^2 + p \; ; p \in GF(2^n) \quad (11)$$

$$P_C(x) = x^4 + x^3 + p \; ; p \in GF(2^n) \quad (12)$$

are called type A, type B, and type C polynomials, respectively.

THEOREM 1 [4]. *Let f be an irreducible polynomial over $GF(2^n)$ of degree m. Let $t \in N$. Then, f factors into d irreducible polynomials in $GF(2^{tn})$ of the same degree m/d, where $d = \gcd(t, m)$.*

LEMMA 3 [19]. *Type A polynomials are irreducible over $GF(2^n)$ if and only if n is odd.*

PROOF. This follows immediately from Theorem 1 and the fact that $P_A(x)$ is irreducible over $GF(2)$. □

In the following, for $\alpha \in GF(2^n)$ let $Tr_{2^n|2}(\alpha)$ denote the *absolute* trace function of $GF(2^n)$ over $GF(2)$, i.e.,

$$Tr_{2^n|2}(\alpha) := \alpha + \alpha^2 + \alpha^{2^2} + \cdots + \alpha^{2^{n-1}}.$$

THEOREM 2 [4]. *For $\alpha, \beta \in GF(2^n)^*$, the trinomial $x^2 + \alpha x + \beta$ is irreducible over $GF(2^n)$ if and only if $Tr_{2^n|2}(\beta/\alpha^2) \neq 0$.*

THEOREM 3. *Let $f(x) = x^m + a_{m-1}x^{m-1} + \cdots + 1$ be an irreducible polynomial over $GF(2^n)$ and let $\beta \in GF(2^n)$. Then, the polynomial $f(x^2 + x + \beta)$ is irreducible over $GF(2^n)$ if and only if $Tr_{2^n|2}(a_{m-1}) \neq 0$.*

LEMMA 4. *Type B polynomials are irreducible over $GF(2^n)$ if and only if*

$$Tr_{2^n|2}(p) = Tr_{2^n|2}(p^{-1}) = 1.$$

In particular, such an element p always exists.

PROOF. According to Theorem 2, the polynomial $f(x) = x^2 + \gamma x + 1$ is irreducible over $GF(2^n)$ if and only if $Tr_{2^n|2}(\gamma^{-2}) = Tr_{2^n|2}(\gamma^{-1})^2 = Tr_{2^n|2}(\gamma^{-1}) = 1$. Suppose now that $Tr_{2^n|2}(\gamma^{-1}) = 1$. Then, from Theorem 3, we obtain that $f(x^2 + x + 1)$ is irreducible over $GF(2^n)$ if and only if $Tr_{2^n|2}(\gamma) = 1$. So, the polynomial

$$g(x) = x^4 + (\gamma + 1)x^2 + \gamma x + \gamma$$

is irreducible over $GF(2^n)$ if and only if $\mathrm{Tr}_{2^n|2}(\gamma^{-1}) = \mathrm{Tr}_{2^n|2}(\gamma) = 1$. Now, take the reciprocal polynomial of $g(x)$, i.e., $x^4 g(1/x)$, multiply it by γ^{-1}, and set $p := \gamma^{-1}$.

To show that such an element p exists, we can assume that n is even (if n is odd, just take $p = 1$). Half of the elements of $GF(2^n)$ have trace 0 and the other half trace 1. If n is even, then $Tr_{2^n|2}(0) = Tr_{2^n|2}(1) = 0$. Now, order the rest of the elements in pairs (p_i, p_i^{-1}), $i = 1, 2, \ldots, 2^{n-1} - 1$. Since we have 2^{n-1} ones left and $2^{n-1} - 2$ zeros, there must be at least one pair (p_i, p_i^{-1}) with $Tr_{2^n|2}(p_2) = Tr_{2^n|2}(p_j^{-1}) = 1$. □

A possible way to find an element $p \in GF(2^n)$ satisfying the trace condition is described in the following lemma. The problem will be reduced to finding a root of a certain polynomial in $GF(2^n)$. For this problem, efficient (probabilistic) algorithms exist (see [19]).

LEMMA 5. *Define polynomials $s_j(x) \in GF(2)[x]$, $j \geq 1$ recursively*

$$s_1(x) = x^2 + x + 1,$$

$$s_{j+1}(x) = x^{2^j} s_j\left(x + \frac{1}{x}\right).$$

Let $n = 2^k m$ with m odd. Then, a root $p \in GF(2^n)$ of the polynomial $s_k(x)$ satisfies $Tr_{2^n|2}(p) = Tr_{2^n|2}(p^{-1}) = 1$.

PROOF. Note that the polynomials $s_j(x)$ are self-reciprocal of degree 2^j and that the second highest (and also the second lowest) coefficient are equal to one. It follows from Theorem 3.10 in [4] or from Theorem 9 in [23] that these polynomials are irreducible over $GF(2)$. If p is a root of $s_k(x) = \sum_{i=0}^{2^k} s_i x^i$, then it follows that

$$Tr_{2^n|2}(p) = mTr_{2^k|2}(p) = ms_{2^k-1} = 1.$$

The same argument holds for p^{-1}, since this is also a root of $s_k(x)$. □

LEMMA 6. *Irreducible type C polynomials exist for all ground fields $GF(2^n)$ if n is odd.*

PROOF. In analogy to the case of type A polynomials, this follows immediately from Theorem 1 and the fact that $P_C(x) = x^4 + x^3 + 1$ is irreducible over $GF(2)$. □

So far, conditions for the irreducibility of the three types of polynomials have been given. In the following, irreducible type A and type C polynomials will be used for odd n, while irreducible type B polynomials will be used for even n. The reason that two different types of polynomials are used for odd n is that type C polynomials are potentially primitive, while type A polynomials are not. Primitivity is sometimes of interest in practice. For most of the (small) fields $GF(2^n)$ of technical interest, we were able to determine primitive ones through computer searches. These will be provided in Table 2.

4.2 Multiplier Architectures

This subsection derives efficient multiplier architectures from the three types of field polynomials. All multipliers are constructed from the first two stages of the KOA given in (5) and mappings of form (9).

4.2.1 Type A Multipliers

Type A multipliers are based on the polynomials $P_A(x)$. Their computational complexity is given by the following theorem:

LEMMA 7. *Type A multiplier in $GF(2^n)^4$) can be realized with a complexity of*

$$\#\oplus = 21, \tag{13}$$

$$\#\otimes = 9, \tag{14}$$

additions and multiplications, respectively, in $GF(2^n)$. The delay T of a parallel realization is given by

$$T = 5T_\oplus + T_\otimes,$$

where "$T_\oplus$" denotes the delay caused by an adder in $GF(2^n)$ and "$T_\otimes$" denotes the delay of a multiplier.

PROOF. The first two stages of the KOA require nine multiplications and 10 additions, as shown in (5). The results of the second stage are nine auxiliary coefficients d_i. If a type A polynomial is used for the construction of the mappings (7) and (9), the coefficients of a product polynomial $C(x) = \sum_{i=0}^{3} c_i x^i$ are obtained as:

$$\begin{aligned} c_0 &= d_0 + d_2 + d_5 + d_7 \\ c_1 &= d_0 + d_1 + d_5 + d_6 \\ c_2 &= d_0 + d_3 + d_5 + d_8 \\ c_3 &= d_0 + d_1 + d_3 + d_4 + d_7. \end{aligned}$$

The set of equations can be realized with 11 additions when the term $(d_0 + d_5)$ is precomputed. The overall complexity is, thus, nine multiplications and 10 + 11 = 21 additions.

From Fig. 1, it follows that the delay introduced by the first two stages of the KOA is $2T_\oplus$ and $1T_\otimes$. Computation of c_3 from the d_is adds another delay of $3T_\oplus$. The over-all critical path thus contains $3 + 2 = 5T_\oplus$ and $1T_\otimes$. □

In order to further illustrate the structure of the multiplier, Fig. 2 shows a block diagram of a parallel hardware realization of a type A multiplier. The circuit uses the two building blocks "$GF(2^n)$ adder" and "$GF(2^n)$ multiplier". The complexities from (13) and (14) can easily be verified from the diagram. The inputs are the variables a_i, b_i, $i = 0, 1, 2, 3$, the outputs are the c_i. It should be noted that the inputs and outputs and all internal connections are n bit wide buses. The figure also underlines the high degree of modularity of the architecture. An entire multiplier in the composite field $GF(2^n)^m$) can be constructed from only two modules which provide $GF(2^n)$ addition and multiplication, respectively.

We now compare the space and time complexity of the type A multiplier with the complexities of the pure polynomial multiplication. From Lemmas 1 and 2, it can be seen that the entire field multiplication, consisting of polynomial multiplication and modulo reduction, can be performed with a complexity *smaller* than the complexity for the polynomial multiplication only. This holds for the computational complexity (number of additions and multiplications) as well as for the delay. We would like to stress that the improved complexities are solely due to a reduced number of additions in the subfield $GF(2^n)$. The number of subfield multiplications is identical for both architectures.

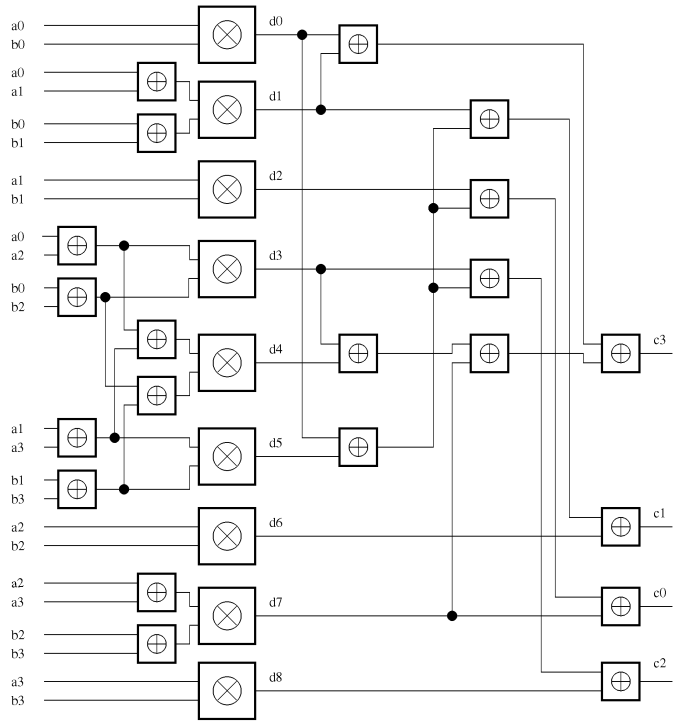

Fig. 2. Block diagram of a parallel realization of a type A multiplier.

4.2.2 Type B Multipliers

Type B multipliers are based on the polynomials $P_B(x)$. Their computational complexity is given by the following lemma:

LEMMA 8. *Type B multiplier in $GF(2^n)^4$) can be realized with a complexity of*

$$\begin{aligned} \#\oplus &= 25, \\ \#\otimes_{\text{cnst}} &= 4, \\ \#\otimes &= 9, \end{aligned} \tag{15}$$

additions, constant multiplications, and general multiplications, respectively, in $GF(2^n)$. The delay of a parallel realization is given by

$$T = 6T_\oplus + T_\otimes + T_{\otimes_p},$$

where "$T_{\otimes_p}$" denotes the delay of multiplication with the coefficient p.

PROOF. We just have to show that the mapping (9) can be done with 15 additions and four constant multiplications. Applying (7) yields the following product coefficients:

$$\begin{aligned} c_0 &= d_0 + p\left(d_2 + d_5 + d_7\right) + p^2 d_8 \\ c_1 &= d_0 + d_1 + d_2 + p\left(d_6 + d_7\right) \\ c_2 &= c_0 + d_3 + d_5 + d_6 + d_7 \\ c_3 &= d_0 + d_1 + p\left(d_6 + d_7\right) + d_3 + d_4 + d_7 + (p+1)d_8. \end{aligned}$$

TABLE 1
FIELD POLYNOMIALS AND COMPLEXITIES OF THREE CLASSES OF MULTIPLIERS IN $GF(2^n)^4)$

	polynomial	exist.	prim.	computational complexity			delay		
				AND	XOR	$\otimes_{\text{cnst}}$	$\mathcal{T}_{\text{and}}$	$\mathcal{T}_{\text{xor}}$	$T_{\otimes_p}$
A	$x^4 + x^3 + x^2 + x + 1$	odd n	no	$9n^2$	$9n^2 + 21n - 9$	0	1	$2\lceil \log_2 n \rceil + 5$	0
B	$x^4 + x^3 + (p+1)x^2 + p$	all n	yes	$9n^2$	$9n^2 + 25n - 9$	4	1	$2\lceil \log_2 n \rceil + 6$	1
C	$x^4 + x^3 + p$	odd n	yes	$9n^2$	$9n^2 + 24n - 9$	3	1	$2\lceil \log_2 n \rceil + 5$	1

The four equations have exactly this computational complexity if c_0, $d_3 + d_7$, and $d_0 + d_1 + p(d_6 + d_7)$ are precomputed.

The computation of c_2 from the d_is requires $4T_\oplus$ and $1T_{\otimes_p}$. The critical path is formed by this delay together with the delay of $2T_\oplus$ and $1T_\otimes$ introduced by the first two stages of the KOA. □

4.2.3 Type C Multipliers

Type C multipliers are based on the polynomials $P_C(x)$. Their computational complexity is given by the following lemma:

Lemma 9. *Type C multiplier in $GF(2^n)^4)$ can be realized with a complexity of*

$$\begin{aligned} \#\oplus &= 24, \\ \#\otimes_{\text{cnst}} &= 3, \\ \#\otimes &= 9, \end{aligned} \tag{16}$$

additions, constant multiplications, and general multiplications, respectively, in $GF(2^n)$. The delay of a parallel realization is given by

$$T = 5T_\oplus + T_\otimes + T_{\otimes_p}.$$

PROOF. As in the proof above, we have to show that the mapping (9) can be done with 14 additions and three constant multiplications. Applying (7) yields the product coefficients:

$$\begin{aligned} c_0 &= d_0 + p(d_2 + d_5 + d_7 + d_8) \\ c_1 &= d_0 + d_1 + d_2 + p(d_6 + d_7) \\ c_2 &= d_0 + d_2 + d_3 + d_6 + pd_8 \\ c_3 &= d_0 + d_1 + d_3 + d_4 + d_6. \end{aligned}$$

The four equations have exactly this computational complexity if $d_0 + d_1$ and $d_3 + d_6$ are precomputed.

The computation of c_3 from the d_is requires $3T_\oplus$ and $1T_{\otimes_p}$. The critical path is formed by this delay together with the delay of $2T_\oplus$ and $1T_\otimes$ introduced by the first two stages of the KOA. □

5 RESULTS

This section will summarize the features of the three types of multipliers. Also, space and time complexity expressions on the gate-level will be developed. For a number of fields of technical interest, optimized polynomials are presented.

5.1 Gate Complexities

In order to obtain gate counts, the three operations in the ground field $GF(2^n)$, addition, multiplication, and constant multiplication, must be further specified. Addition requires n XOR gates. General multiplication requires n^2 AND gates and $n^2 - 1$ XOR gates for most small fields of technical interest. In particular, this complexity is achieved for $n = 2, 3, \ldots, 7$ [20]. The complexity of constant multiplication "$\otimes_{\text{cnst}}$" is heavily dependent on the actual field element multiplied with. The average complexity is given by $(n^2/2) - n$ XOR gates [21]. However, by carefully choosing polynomials with suited coefficients, the constant multiplication complexity can be considerably below the average [24], [26]. Polynomials optimized this way are listed in Table 2. The complexity contribution stemming from the constant multiplications is extremely small in all cases. If the constant multiplications are neglected, our architecture asymptotically improves the $k^2 = 16n^2$ complexity bound for traditional architectures by 44 percent.

The architecture limits the speed of multiplier implementations by the delay through the critical path. Since adders in $GF(2^n)$ are realized by n parallel XORs, the delay of an adder is one XOR gate delay. A multiplier in $GF(2^n)$ is upper bounded by

$$T \le \mathcal{T}_{\text{and}} + 2\mathcal{T}_{\text{xor}}\lceil \log_2 n \rceil, \tag{17}$$

where $\mathcal{T}_{\text{and}}$ and $\mathcal{T}_{\text{xor}}$ are the delays of one AND gate and one XOR gate, respectively.

Table 1 lists, for each of the three types of polynomials, the condition for existence, a statement regarding primitivity, the gate complexity, and the number of gate delays along the critical path. The "prim." column provides information about whether the polynomial is potentially primitive.

5.2 Optimized Examples and Comparison

For many applications, such as Reed-Solomon coders, it is helpful to have type B and C polynomials which fulfill the following criteria:

1) The coefficient p is optimized with respect to the complexities of the constant multiplications occurring in (15) and (16).
2) The polynomials are primitive.

We performed exhaustive searches according to the two optimization criteria for all ground fields $GF(2^n)$, $n = 2, 3, \ldots, 8$. The field polynomials used for the ground fields are listed in the column headed by "$Q(y)$" in Table 2. The polynomials are given by their binary coefficients, highest

TABLE 2
OPTIMIZED MULTIPLIERS IN FIELDS $GF(2^k)$, WHERE $k = 4n$

k	n	$Q(y)$	$P(x)$	type	prim.	$\mathcal{T}_{and}$	$\mathcal{T}_{xor}$	AND	XOR	[20], [21] AND	[20], [21] XOR
8	2	111	$11\omega^2 0\omega$	B	yes	**1**	**9**	**36**	**81**	64	84
12	3	1011	11111	A	no	**1**	**8**	**81**	**135**	144	207
			1100ω	C	yes	**1**	**9**	**81**	**147**		
16	4	10011	$11\omega^{14}0\omega^3$	B	no	**1**	**10**	**144**	**246**	256	281
20	5	100101	11111	A	no	**1**	**10**	**225**	**321**	400	399
			$1100\omega^{30}$	C	yes	**1**	**11**	**225**	**339**		
24	6	1000011	$11\omega^{60}0\omega^{29}$	B	yes	**1**	**12**	**324**	**489**	576	> 576
28	7	10000011	11111	A	no	**1**	**9**	**441**	**579**	784	783
			1100ω	C	yes	**1**	**10**	**441**	**602**		
32	8	100101101	$11\omega^3 0\omega^{38}$	B	yes	**1**	**13**	**576**	**993**	1024	> 1024

coefficient leftmost. All of the $Q(y)$ polynomials are primitive. For the fields with even n, optimized type B polynomials were determined. In all but the case $n = 4$, primitive polynomials were found. Among the existing primitive polynomials, the one yielding the lowest constant multiplication complexity was chosen. The corresponding coefficients are provided in an exponent notation "ω^i" in Table 2, where ω is a primitive element such that $Q(\omega) = 0$. For fields with n odd, primitive type C polynomials were determined by the exhaustive search. Again, for each given n, the polynomial with the lowest constant multiplication complexity was chosen.

Table 2 gives detailed information about the gate complexity of multipliers in $GF((2^n)^4) \cong GF(2^k)$, where $k = 4n$. The multipliers are based on the optimized polynomials listed in the column headed by "$P(x)$." Each entry contains the five coefficients of the polynomial, highest coefficient leftmost. In order to allow for comparison with traditional multipliers which have a gate count of at least $k^2 = (4n)^2$ AND gates and $k^2 - 1$ XOR gates, the complexity of the architecture, from [20], [21], based on primitive polynomials is included. It should be noted that traditional architectures reach the $k^2 - 1$ bound only if an irreducible trinomial of degree k exists. This is, however, not true for all fields where $8|k$, which, unfortunately, includes many fields of practical interest. The two rightmost columns contain the number of AND gates and XOR gates along the critical path. It should be noted that all complexities are actual gate counts and delays based on the ground field multiplier [20] and the optimized constant multipliers described in [24].

EXAMPLE. We consider the multiplier in $GF((2^3)^4) \cong GF(2^{12})$ based on a type C polynomial. The ground field $GF(2^3)$ has the primitive field polynomial $Q(y) = y^3 + y + 1$. The extension field has the primitive field polynomial $P(x) = x^4 + x^3 + \omega$, where $Q(\omega) = 0$. Each of the nine multipliers in $GF(2^3)$ has a gate count of nine AND and eight XOR gates [20]. Each ground field adder requires three XOR gates. Constant multiplication with $p = \omega$ requires one XOR gate. Hence, the overall complexity is $9 \cdot 9 = 81$ AND gates and $9 \cdot 8 + 24 \cdot 3 + 3 \cdot 1 = 147$ XOR gates.

The critical path consists of five ground field adders, one ground field multiplier and one constant multiplier with ω. Each adder introduces one XOR gate delay, the general multiplier three XOR and one AND gate delays, and the constant multiplier one XOR gate delay. Hence, the over-all delay results in nine $\mathcal{T}_{xor}$ and one $\mathcal{T}_{and}$.

It is interesting to compare the time and space complexities with other architectures. The traditional architecture [20] for $GF(2^{12})$ requires considerably more gates; our architecture shows an improvement of 44 percent with respect to the AND gate count and 29 percent XOR with respect to the XOR gate count. The delay of the traditional architecture, however, is slightly smaller; seven $\mathcal{T}_{xor}$ as opposed to nine $\mathcal{T}_{xor}$. Comparing the multiplier to the one based on the standard KOA with separate modulo reduction [25], as described in Section 2, it can be seen that the number of XOR gates is reduced from 159 to 135. However, the main advantage seems to be that the delay is reduced from 11 $\mathcal{T}_{xor}$ to eight $\mathcal{T}_{xor}$, or 27 percent, which can be a major factor for applications with high speed requirements.

5.3 Applications

There are two main area of applications for Galois field arithmetic. Certain error correction codes, in particular the popular Reed-Solomon codes, are based on arithmetic in finite fields of characteristic two. The specific field $GF(2^8)$ has been standardized for space communication by ESA and NASA [18], and for use in CD players. Since the latter one constitutes a high volume application, provision of highly optimized architecture is attractive. We would like to stress that the gate count achieved by the new architecture for $GF((2^2)^4) \cong GF(2^8)$ shows an improvement of 44 percent for the AND gate count, together with a slightly improved XOR gate count compared to traditional approaches. Application of larger Galois fields to Reed-Solomon codes is also possible. The results in Table 2 provide

detailed information for designer of corresponding implementations.

The other main area of application for Galois field arithmetic are public-key cryptographic schemes. For our architectures schemes, with moderate field order and the possibility of composed extension degrees, $k = n \cdot m$ are of importance. Elliptic curve [16] and hyperelliptic curve [17] cryptosystems fulfill both requirements. The latter one can be based on field orders as small as $k \geq 30$, whereas the former one requires values in the range of $k = 150 - 250$. In principle, our architectures are applicable to both schemes.

6 Conclusion

New multiplier architectures for composite Galois fields of the form $GF(2^n)^4)$ have been considered. It was shown that the combination of the Karatsuba-Ofman algorithm and the modulo reduction can lead to a reduced number of computations required for a field multiplication. The computational gain is heavily dependent on the field polynomial used. By means of exhaustive searches, we determined three classes of polynomials which lead to optimum multipliers with respect to computational complexity. For each field polynomial type, conditions of existence were provided. Ways of finding these polynomials were provided, too.

Our approach asymptotically improves the $k^2 = 16n^2$ complexity bound for traditional architectures by 44 percent. However, considerable improvements are given for small fields which are of primary technical interest, too. For the fields where $n = 2, 3, \ldots, 8$, field polynomials with optimized coefficients were determined. For each field, except for $n = 4$, primitive polynomials are provided. The exact gate complexity and delays are given for each multiplier. A comparison with multipliers over fields $GF(2^k)$, $k = n\ m$, shows that the new architectures lead to a considerably improved gate count, whereas the theoretical delay of the new architecture is only slightly larger. Compared to composite field architectures which separate polynomial multiplication and modulo reduction [25], the new approach shows a reduced delay together with a further improved gate count.

The new architectures are naturally modular, which is a very attractive feature for VLSI design and implementation. A multiplier can be realized by using only a small number of modules providing $GF(2^n)$ arithmetic. The architectures have applications in the area of error correction codes as well as in public-key cryptography.

References

[1] V.B. Afanasyev, "Complexity of VLSI Implementation of Finite Field Arithmetic," *Proc. II. Int'l Workshop Algebraic and Combinatorial Coding Theory*, pp. 6-7, Leningrad, Sept. 1990.

[2] V.B. Afanasyev, "On the Complexity of Finite Field Arithmetic," *Proc. Fifth Joint Soviet-Swedish Int'l Workshop Information Theory*, pp. 9-12, Moscow, Jan. 1991.

[3] R.E. Blahut, *Fast Algorithms for Digital Signal Processing*. Reading, Mass.: Addison-Wesley, 1985.

[4] A. Menezes, I. Blake, X. Gao, R. Mullin, S. Vanstone, and T. Yaghgoobin. *Applications of Finite Fields*. Kluwer Academic Publisher, 1993.

[5] S.T.J. Fenn, M. Benaissa, and D. Taylor, "$GF(2^m)$ Multiplication and Division over the Dual Base," *IEEE Trans. Computers*, vol. 45, no. 3, pp. 319-327, Mar. 1996.

[6] W. Geiselmann, "Algebraische Algorithmenentwicklung am Beispiel der Arithmetik in Endlichen Körpern," PhD thesis, Universität Karlsruhe, Fakultät für Informatik, Institut für Algorithmen und Kognitive Systeme, Karlsruhe, Germany, 1993.

[7] D.H. Green and I.S. Taylor, "Irreducible Polynomials over Composite Galois Fields and Their Applications in Coding Techniques," *Proc. IEE*, vol. 121, no. 9, pp. 935-939, Sept. 1974.

[8] M.A. Hasan, "Efficient Computations in Galois Fields," PhD thesis, Dept. of Electrical and Computer Eng., Univ. of Victoria, Canada, Apr. 1992.

[9] M.A. Hasan, M. Wang, and V.K. Bhargava, "Division and Bit-Serial Multiplication over $GF(q^m)$," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 972-980, Aug. 1992.

[10] M.A. Hasan, M. Wang, and V.K. Bhargava, "Modular Construction of Low Complexity Parallel Multipliers for a Class of Finite Fields $GF(2^m)$," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 962-971, Aug. 1992.

[11] 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 Trans. Computers*, vol. 37, no. 6, pp. 735-739, June 1988.

[12] T. Itoh and S. Tsujii, "Structure of Parallel Multipliers for a Class of Fields $GF(2^k)$," *Information and Computers*, vol. 83, pp. 21-40, 1989.

[13] Y. Jeong, "VLSI Algorithms and Architectures for Real-Time Computation over Finite Fields," PhD thesis, Dept. of Electrical and Computer Eng., Univ. of Massachusetts at Amherst, Feb. 1995.

[14] A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata," *Sov. Phys.-Dokl. (English translation)*, vol. 7, no. 7, pp. 595-596, 1963.

[15] D.E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, second ed. Reading, Mass.: Addison-Wesley, 1981.

[16] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Computation*, vol. 48, pp. 203-209, 1987.

[17] N. Koblitz, "Hyperelliptic Cryptosystems," *J. Cryptology*, vol. 1, no. 3, pp. 129-150, 1989.

[18] H. Kummer, "Recommendation for Space Data System Standards: Telemetry Channel Coding: Issue-1," Consult. Comm. Space Data Systems, Sept. 1983.

[19] R. Lidl and H. Niederreiter, *Finite Fields*, vol. 20, *Encyclopedia of Math. and Its Applications*. Reading, Mass.: Addison-Wesley, 1983.

[20] E.D. Mastrovito, "VLSI Design for Multiplication over Finite Fields $GF(2^m)$," *Lecture Notes in Computer Science 357*, pp. 297-309. Berlin: Springer-Verlag, Mar. 1989.

[21] E.D. Mastrovito, "VLSI Architectures for Computation in Galois Fields," PhD thesis, Dept. of Electrical Eng., Linköping Univ., Linköping, Sweden, 1991.

[22] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.

[23] H. Meyn, "On the Construction of Irreducible Self-Reciprocal Polynomials over Finite Fields," *Applicable Algebra Eng., Comm., and Computing*, vol. 1, no. 1, pp. 43-53, 1990.

[24] C. Paar, "Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields," PhD thesis, (English translation), Inst. for Experimental Math., Univ. of Essen, Essen, Germany, June 1994.

[25] C. Paar, "A New Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Fields," *IEEE Trans. Computers*, vol. 45, no. 7, pp. 856-861, July 1996.

[26] C. Paar, "Optimized Arithmetic for Reed-Solomon Encoders," *Proc. 1997 IEEE Int'l Symp. Information Theory*, p. 250, Ulm, Germany, June 29–July 4, 1997.

[27] C. Paar and N. Lange, "A Comparative VLSI Synthesis of Finite Field Multipliers," *Proc. Third Int'l Symp. Comm. Theory and Its Applications*, Lake District, U.K., July 10-14, 1995.

[28] A. Pincin, "A New Algorithm for Multiplication in Finite Fields," *IEEE Trans. Computers*, vol. 38, no. 7, pp. 1,045-1,049, July 1989.

[29] P.A. Scott, S.E. Travares, and L.E. Peppard, "A Fast VLSI Multiplier for $GF(2^m)$," *IEEE J. Selectected Areas Comm.*, vol. 4, pp. 62-66, Jan. 1986.

[30] C.C. Wang and D. Pei, "A VLSI Design for Computing Exponentiation in $GF(2^m)$ and Its Application to Generate Pseudorandom Number Sequences," *IEEE Trans. Computers*, vol. 39, no. 2, pp. 258-262, Feb. 1990.

[31] C.C. Wang, T.K. Truong, H.M. Shao, L.J. Deutsch, J.K. Omura, and I.S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in $GF(2^m)$," *IEEE Trans. Computers*, vol. 34, no. 8, pp. 709-717, Aug. 1985.

[32] *Reed-Solomon Codes and Their Applications*, S.B. Wicker and V.K. Bhargava, eds. IEEE Press, 1994.

Christof Paar received his first degrees from the Fachhochschule in Cologne and the University of Siegen in 1988 and 1991, respectively. From 1991 to 1994, he was with the Institute for Experimental Mathematics, University of Essen, where he received a PhD in electrical engineering. Since 1995, he has been an assistant professor at Worcester Polytechnic Institute, Worcester, Massachusetts, where he leads the Cryptography and Information Security Group. Dr. Paar's research interests include efficient hardware and software algorithms for public-key systems, FPGAs in cryptography, and implementation of elliptic and hyperelliptic curve algorithms. He received a U.S. National Science Foundation CAREER award for investigating cryptographic algorithms on FPGAs. He is a member of the IEEE, ACM, and IACR.

Peter Fleischmann received his PhD in mathematics from the University of Essen in 1986. He is member of the Institute for Experimental Mathematics at the University of Essen. He is an algebraist working in finite groups and representation theory. He is also working in applied algebra, in particular in the theory and applications of finite fields. He is a member of the American Mathematical Society.

Peter Roelse received his Ir degree (cum laude) in applied mathematics in 1994 from the Delft University of Technology, The Netherlands. In the same year, he joined the Algebra Group of the Institute for Experimental Mathematics, University of Essen, Germany. He received his PhD in mathematics in September 1997. His main research interests are coding theory, cryptography, and algorithmic and computational aspects of finite fields, including their efficient software implementations.