Finite Field Multiplier Using Redundant Representation

* Huapeng Wu, † M. Anwar Hasan, ‡ Ian F. Blake, § Shuhong Gao

January 3, 2002

Abstract

This article presents simple and highly regular architectures for finite field multipliers using a redundant representation. The basic idea is to embed a finite field into a cyclotomic ring which has a basis with the elegant multiplicative structure of a cyclic group. One important feature of our architectures is that they provide area-time trade-offs which enable us to implement the multipliers in a partial-parallel/hybrid fashion. This hybrid architecture has great significance in its VLSI implementation in very large fields. The squaring operation using the redundant representation is simply a permutation of the coordinates. It is shown that when there is an optimal normal basis, the proposed bit-serial and hybrid multiplier architectures have very low space complexity. Constant multiplication is also considered and is shown to have advantage in using the redundant representation.

Index terms:

Finite field arithmetic, cyclotomic ring, redundant set, normal basis, multiplier, squaring.

---

*H. Wu is with the Centre for Applied Cryptographic Research, University of Waterloo, Waterloo, Canada. E-mail:h3wu@cacr.math.uwaterloo.ca.

†M. A. Hasan is with the Department of Electrical and Computer Engineering, University of Waterloo. E-mail: ahasan@ece.uwaterloo.ca.

‡I. F. Blake is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada. E-mail: ifblake@comm.utoronto.ca.

§S. Gao is with the Department of Mathematical Sciences, Clemson University. E-mail: sgao@math.clemson.edu. Gao was supported in part by NSF grant #DMS9970637, NSA grant #MDA904-00-1-0048 and ONR grant #N00014-00-1-0565.

1 INTRODUCTION

Efficient computations in finite fields and their architectures are important in many applications including coding theory, computer algebra systems and public-key cryptosystems (*e.g.*, elliptic curve cryptosystems). Although all finite fields of the same cardinality are isomorphic, their arithmetic efficiency depends greatly on the choice of bases for field element representations. The most commonly used bases are polynomial bases (PB) and normal bases (NB), sometimes combined with dual bases (DB)[15]. A major advantage of normal bases in the fields of characteristic two is that the squaring operation in NB is simply a cyclic shift of the coordinates of elements, so these are useful for computing large exponentiations and multiplicative inverses [13, 11, 1]. Also, the multiplication table of a normal basis is symmetric, so suitable for hardware implementation. This is the basis for the multiplier of Massey-Omura [16] and that of Onyszchuk et al. [18].

Recently, Gao et al. [7, 8] have proposed a novel method to perform fast multiplication with a normal basis generated by Gauß periods. The main idea is to embed a field in a larger ring, perform multiplication (using the Fast Fourier Transform) there and then convert the result back to the field. The ring they use is referred to as a *cyclotomic ring* which has an extremely simple basis whose elements form a cyclic group. One purpose of this paper is to make this idea more explicit and present architectures that are suitable for hardware implementation.

We are mainly interested in finite fields of characteristic two, i.e. $\mathbb{F}_{2^m}$, which are one of the two types of fields used most commonly in practice (the other one is $\mathbb{F}_p$ where p is a prime). We show how to find the *smallest* cyclotomic ring in which $\mathbb{F}_{2^m}$ can be embedded. Since "embedding" is *not unique*, each element in the ring can be represented in more than one way, *i.e.*, the representation contains certain amount of redundancy. In this article, we also discuss how this redundant representation of a field element can be efficiently converted to a normal basis and vice versa.

Another purpose of our paper is to present architectures for arithmetic in $\mathbb{F}_{2^m}$. Both bit-serial and hybrid multipliers using the redundant representation are proposed and their complexities are discussed. A modified form of the multipliers using the redundant representation with reduced complexity are also presented. The bit-serial and hybrid architectures of this modified multiplier have lower complexity compared to the previously reported normal basis multipliers. A constant multiplier using the redundant representation is also considered.

We should mention other related work here. Itoh and Tsujii [14] constructed a multiplier for a class of fields defined by irreducible all-one-polynomials (AOPs) and equally-spaced-polynomials (ESPs). Wolf [22] found a simple multiplication architecture for irreducible AOP's. Drolet [4] uses maximum subfields in cyclotomic rings. Silverman [19] considered a special case when there is a type I optimal normal basis. This case is also considered in [7, 8]. A more recent article on redundant representation is [10].

The organization of this paper is as follows: Section 2 shows how redundant representation of a field element can be derived from cyclotomic rings. In Section 3, multiplication operation using the redundant representation is discussed and then basis conversions are given. Architectures of bit-serial, bit-parallel, hybrid, and constant multipliers are presented in Section 4. For the field which has a type II ONB, we show in Section 5 that more efficient architectures can be developed using a basis derived from the redundant representation. This multiplier architecture is highly regular and also has low complexity. Finally, a few concluding remarks are given in Section 6.

2 CYCLOTOMIC FIELDS AND REDUNDANT REPRESENTATION

Let K be any field and n a positive integer. The n-th *cyclotomic field*, denoted by $K^{(n)}$, over K is defined to be the splitting field of $x^n - 1$ over K. In particular, n divides $\#K^{(e)} - 1$ for some e and is thus coprime to the characteristic. Let β be a primitive n-th root of unity in some extension of K. Then $K^{(n)}$ is generated by β over K and elements of $K^{(n)}$ can be written in the form

$$A = a_0 + a_1\beta + a_2\beta^2 + \cdots + a_{n-1}\beta^{n-1}, \quad a_i \in K. \tag{1}$$

Here the representation is not unique, that is, each n-tuple $(a_0, a_1, \ldots, a_{n-1})$, $a_i \in K$, gives an element of $K^{(n)}$ but different tuples may give the same element. For example, since $1 + \beta + \beta^2 + \cdots + \beta^{n-1} = 0$, the two n-tuples $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$ both represent 0, while $(-1, 0, \ldots, 0)$ and $(0, 1, \ldots, 1)$ both represent -1. Because of such redundant representations, and by a slight abuse of terminology, we denote $[1, \beta, \beta^2, \ldots, \beta^{n-1}]$[1] as a redundant basis for

[1] An *ordered* set of field elements is denoted by $[\cdots]$.

any subfield of $K^{(n)}$ containing K. Note that a redundant basis is unique up to order for a given $K^{(n)}$.

On the other hand, we may consider the ring $K[x]/(x^n-1)$, called the n-th *cyclotomic ring*, denoted by $R_n(K)$. If we let β be the congruence class of x, then $\beta^n \equiv 1$ and elements of R_n can also be represented in the form (1). But now the representation is *unique* and so the elements $1, \beta, \beta^2, \ldots, \beta^{n-1}$ form a *true basis* for R_n. Note that the elements $1, \beta, \beta^2, \ldots, \beta^{n-1}$ form a cyclic group of order n and

$$\beta \cdot \beta^i = \begin{cases} \beta^{i+1} & i \neq n-1, \\ 1 & i = n-1. \end{cases} \tag{2}$$

This simple multiplication table allows us to design efficient architectures of low complexity as shown in Section 3.

Suppose that $\mathbb{F}_{q^m}$ is embedded in $\mathbb{F}_q^{(n)}n$, where q is a prime power. Then arithmetic in $\mathbb{F}_{q^m}$ using the redundant representation can be performed following these three steps:

1. Represent elements in $\mathbb{F}_{q^m}$ in the form (1);
2. View them in the ring R_n and do arithmetic there;
3. Finally convert the result back to $\mathbb{F}_{q^m}$.

We characterize here all the fields that can be embedded in $K^{(n)}$ when $K = \mathbb{F}_q$.

Theorem 1 [15] *Let q be a prime power and n be a positive integer with $\gcd(q, n) = 1$. Then $\mathbb{F}_{q^m}$ is contained in $\mathbb{F}_q^{(n)}$ iff m divides the multiplicative order of q modulo n.*

Proof: Let d be the multiplicative order of q modulo n. By Theorem 2.47 (page 65) of [15], $\mathbb{F}_q^{(n)}$ has degree d so it is isomorphic to $\mathbb{F}_{q^d}$. The theorem follows, as $\mathbb{F}_{q^m}$ is contained in $\mathbb{F}_{q^d}$ if and only if $m \mid d$. $\square$

Remark 1 *If there is a type I optimal normal basis in $\mathbb{F}_{2^m}$ then $\mathbb{F}_{2^m}$ is contained in $\mathbb{F}_2^{(m+1)}$, so there is a redundant basis of size $m+1$ for $\mathbb{F}_{2^m}$.*

Here a basis for $\mathbb{F}_{2^m}$ is $\{\beta, \beta^2, \ldots, \beta^m\}$ and the correspondence between field elements and ring elements is

$$\begin{aligned} a_1\beta + a_2\beta^2 + \cdots + a_m\beta^m &\mapsto 0 \cdot 1 + a_1\beta + a_2\beta^2 + \cdots + a_m\beta^m \\ (a_1 + a_0)\beta + (a_2 + a_0)\beta^2 + \cdots + (a_m + a_0)\beta^m &\hookleftarrow a_0 \cdot 1 + a_1\beta + a_2\beta^2 + \cdots + a_m\beta^m. \end{aligned}$$

This is the case considered by Silverman, Gao, et al. [19, 7, 8].

Remark 2 *If there is a type II optimal normal basis in $\mathbb{F}_{2^m}$ then $\mathbb{F}_{2^m}$ is contained in $\mathbb{F}_2^{(2m+1)}$, so there is a redundant basis of size $2m+1$ for $\mathbb{F}_{2^m}$.*

This case will be considered in more detail in Section 5. In concluding this section, in Table 1 we give the smallest values of n for $151 \leqslant m \leqslant 250$ such that $\mathbb{F}_{2^m}$ is contained in $\mathbb{F}_2^{(n)}$.

3 MULTIPLICATION USING REDUNDANT REPRESENTATION

From now on we only consider fields of characteristic two.

3.1 Multiplication Operation

Consider the basis of our redundant representation for $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$:

$$I_1 = [1, \beta, \beta^2, \ldots, \beta^{n-1}].$$

Let any two field elements $A, B \in \mathbb{F}_{2^m}$ be represented with respect to (w.r.t.) I_1, *i.e.*, $A = \sum_{i=0}^{n-1} a_i\beta^i$ and $B = \sum_{i=0}^{n-1} b_i\beta^i$, where $a_i, b_i \in \mathbb{F}_2$ are the coordinates of A w.r.t. I_1. Note that $n \geqslant m+1$ and the lists of coefficients $\{a_i\}$ and $\{b_i\}$ are not unique.

Since $\beta^n = 1$, the product of field elements A and B can be given by

$$AB = \sum_{i=0}^{n-1} a_i(\beta^i \cdot B) = \sum_{i=0}^{n-1} a_i(\sum_{j=0}^{n-1} b_j\beta^{i+j}) = \sum_{i=0}^{n-1} a_i(\sum_{j=0}^{n-1} b_{(j-i)}\beta^j) = \sum_{j=0}^{n-1} \Big(\sum_{i=0}^{n-1} a_i b_{(j-i)}\Big)\beta^j,$$

where $(j-1)$ in the subscript denotes that $j-1$ is to be reduced modulo n.

m	n	n/m	m	n	n/m	m	n	n/m	m	n	n/m
151	907	6.0	176	1409	8.0	201	1609	8.0	226	227	1.0
152	1217	8.0	177	709	4.0	202	809	4.0	227	5449	24.0
153	613	4.0	178	179	1.0	203	841	4.1	228	1603	7.0
154	617	4.0	179	359	2.0	204	409	2.0	229	2749	12.0
155	311	2.0	180	181	1.0	205	821	4.0	230	461	2.0
156	169	1.1	181	1087	6.0	206	619	3.0	231	463	2.0
157	1571	10.0	182	547	3.0	207	829	4.0	232	929	4.0
158	317	2.0	183	367	2.0	208	2081	10.0	233	467	2.0
159	749	4.7	184	799	4.3	209	419	2.0	234	1007	4.3
160	2123	13.3	185	1481	8.0	210	211	1.0	235	941	4.0
161	967	6.0	186	373	2.0	211	2111	10.0	236	709	3.0
162	163	1.0	187	1123	6.0	212	535	2.5	237	1423	6.0
163	653	4.0	188	941	5.0	213	853	4.0	238	717	3.0
164	415	2.5	189	379	2.0	214	643	3.0	239	479	2.0
165	661	4.0	190	573	3.0	215	1291	6.0	240	1067	4.4
166	499	3.0	191	383	2.0	216	1297	6.0	241	1447	6.0
167	2339	14.0	192	769	4.0	217	1303	6.0	242	1331	5.5
168	833	5.0	193	773	4.0	218	1091	5.0	243	487	2.0
169	677	4.0	194	389	2.0	219	877	4.0	244	733	3.0
170	1021	6.0	195	869	4.5	220	575	2.6	245	491	2.0
171	361	2.1	196	197	1.0	221	443	2.0	246	581	2.4
172	173	1.0	197	3547	18.0	222	1043	4.7	247	1483	6.0
173	347	2.0	198	437	2.2	223	2677	12.0	248	1489	6.0
174	349	2.0	199	797	4.0	224	449	2.0	249	1169	4.7
175	701	4.0	200	401	2.0	225	1919	8.5	250	625	2.5

Table 1: Smallest cyclotomic field ${\mathbb{F}_2}^{(n)}$ that contains $\mathbb{F}_{2^m}$ as a subfield.

If we define $AB = C \triangleq \sum_{j=0}^{n-1} c_j \beta^j$, we have

$$c_j = \sum_{i=0}^{n-1} a_i b_{(j-i)}, \quad j = 0, 1, \ldots, n-1. \tag{3}$$

Then a multiplication operation using the redundant representation is decided by expression (3). On the other hand, the squaring of an element A using basis I_1 can simply be performed as follows:

$$A^2 = a_0 + a_1\beta^2 + \cdots + a_{n-1}\beta^{2(n-1)}.$$

Since $\beta^n = 1$, we have $\beta^{2j} = \beta^{2j-n}$ if $2j > n-1$. Note that n is odd because of the minimum of the redundant basis, thus A^2 can be written as

$$\begin{aligned} A^2 &= a_0 + a_1\beta^2 + \cdots + a_{\frac{n-1}{2}}\beta^{n-1} + a_{\frac{n+1}{2}}\beta + a_{\frac{n+3}{2}}\beta^3 + \cdots + a_{n-1}\beta^{n-2} \\ &= a_0 + a_{\frac{n+1}{2}}\beta + a_1\beta^2 + a_{\frac{n+1}{2}+1}\beta^3 + \cdots + a_{\frac{n+1}{2}+\frac{n-3}{2}}\beta^{n-2} + a_{\frac{n-1}{2}}\beta^{n-1}. \end{aligned}$$

Clearly, a squaring operation using redundant representation is equivalent to a permutation of the element coordinates.

3.2 Gauß Periods, Normal Bases and Redundant Bases

Some redundant bases can be easily introduced by the normal bases generated with the Gauß period, and by doing so one can find the relation/conversion between the redundant basis and the normal basis. This is discussed below.

The Gauß periods, which were discovered by Gauß, are defined as follows: Let $m, k \geqslant 1$ be integers such that $n = mk + 1$ is a prime, and let q be a prime power with $\gcd(q, n) = 1$. Let $\mathcal{K}$ be the unique subgroup of order k of the multiplicative group $\mathbb{Z}_n^\times$ of $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$, then for any primitive nth root β of unity in $\mathbb{F}_{q^{mk}}$, the element

$$\gamma = \sum_{\alpha\in\mathcal{K}} \beta^\alpha \tag{4}$$

is called a *Gauß period of type* (m, k) over $\mathbb{F}_q$, where α is a kth root of unity in $\mathbb{Z}_{km+1}^\times$. It can be checked that $\gamma \in \mathbb{F}_{q^m}$. For example, when $k = 2$, α is a square root of unity in $\mathbb{Z}_{2m+1}^\times$. So, $\alpha = \pm 1$, and $\gamma = \beta + \beta^{-1}$. This is the case which will be discussed in Section 5.

Gauß periods have been used to construct normal bases with low complexity [17, 3]. (For a definition of the complexity of a normal basis the reader is referred to [3].) A Gauß period of type (m, k) over $\mathbb{F}_2$ naturally introduces a normal basis $I_2 = [\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}]$ in $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ if and only if $\gcd(e, m) = 1$, where e is the order of 2 modulo n. Furthermore, such a normal basis has complexity at most $mk' - 1$ with $k' = k$ if k even and $k + 1$ otherwise [3, 21, 6]. Gauß periods of type $(m, 1)$ and $(m, 2)$ generate optimal normal bases (ONBs) with complexity $2m - 1$, which are usually called type-I and type-II ONBs, respectively [17].

For a normal basis generated with Gauß period of type (m, k), from (4) we have

$$I_2 = [\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}] = [\sum_{i=0}^{k-1} \beta^{\alpha^i}, \sum_{i=0}^{k-1} \beta^{2\alpha^i}, \ldots, \sum_{i=0}^{k-1} \beta^{2^{m-1}\alpha^i}],$$

where α is a primitive kth root of unity in $\mathbb{Z}_{km+1}^{\times}$. Note that each element in I_2 is a sum of k elements. Let the set of these km elements be denoted as $S_1 = \{\beta^{2^i\alpha^j}, i = 0, 1, \ldots, m-1; j = 0, 1, \ldots, k-1\}$. Consider another set of km elements: $S_2 = \{\beta, \beta^2, \ldots, \beta^{km}\}$. It can be seen that elements in S_1 can serve as a "representation basis" for $\mathbb{F}_{2^m}$. Indeed for any element $\beta^{2^i\alpha^j} \in S_1$, we have $\beta^{2^i\alpha^j} = \beta^{2^i\alpha^j \bmod (mk+1)} \in S_2$, and thus, $S_1 \subseteq S_2$. Let $G = \mathbb{Z}_{km+1}^{\times}$ then $G = \langle 2, \alpha \rangle$. For any integer $l \in \{1, 2, \ldots, km\}$, there exist integers $i \in \{0, 1, \ldots, m-1\}$ and $j \in \{0, 1, \ldots, k-1\}$, such that $l = 2^i\alpha^j \bmod (km+1)$. Therefore, $S_2 \subseteq S_1 \Rightarrow S_2 = S_1$. Obviously, besides element "1", the basis of our redundant representation contains exactly the same km elements as S_1 or S_2.

3.3 Conversions of Bases

Among the three steps of redundant representation arithmetic, the first and the final steps deal with the change of representations. In this subsection we discuss the conversions between the normal basis and the redundant basis derived from the Gauß period. We show that such conversions can be done in hardware with almost no cost.

Before giving the conversions between normal basis I_2 and redundant basi I_1, we first introduce two intermediate "bases". Following the discussion in the previous subsection, we separate each sum of k terms of I_2 and put the km elements in an ordered set and let it be denoted by I_3:

$$I_3 = [\beta, \beta^{\alpha}, \ldots, \beta^{\alpha^{k-1}}, \beta^2, \beta^{2\alpha}, \ldots, \beta^{2\alpha^{k-1}}, \ldots, \beta^{2^{m-1}}, \beta^{2^{m-1}\alpha}, \ldots, \beta^{2^{m-1}\alpha^{k-1}}].$$

Clearly, I_3 can serve as a "basis" of $\mathbb{F}_{2^m}$. The second intermediate "basis" is given by

$$I_4 = [\beta, \beta^2, \beta^3, \ldots, \beta^{mk}].$$

From the discussion in the previous subsection, we know that I_4 has exactly the same mk elements as I_3 but with a different order. Moreover, the permutation can be carried out as follows. Let $A \in \mathbb{F}_{2^m}$ and $A = (a_1^{(j)}, a_2^{(j)}, \ldots, a_{mk}^{(j)})$ w.r.t. I_j for $j = 3, 4$. For any $i, 1 \leq i \leq mk$, write

$i = lk + d$, where $1 \le d \le k$ and $0 \le l \le m-1$. Then

$$a_i^{(3)} = a_{lk+d}^{(3)} = a_{(2^l\alpha^{d-1})}^{(4)}, \tag{5}$$

where $(2^l\alpha^{d-1})$ denotes $2^l\alpha^{d-1}$ to be reduced modulo n. In this way, we create a one-to-one correspondence between the I_3 and I_4 based coordinates.

Obviously, conversions between the normal basis and the redundant basis can be divided into three steps:

(a) Conversions between the normal basis and the intermediate basis I_3;

(b) Conversions between two intermediate bases I_3 and I_4;

(c) Conversions between I_4 and the redundant basis

Step (b) has been solved in (5). It can be implemented as a rewiring of lines and has almost no cost in hardware. Step (c) is even simpler. Note that the redundant basis can be obtained by including the element "1" before the first element of I_4. If we let $A = (a_0, a_1, \ldots, a_{mk})$ w.r.t. the redundant representation, then

$$a_i = a_i^{(4)} \text{ for } i = 1, 2, \ldots, mk \text{ and } a_0 = 0. \tag{6}$$

Conversely, if a_i's are given, then

$$a_i^{(4)} = \begin{cases} a_i & \text{if } a_0 = 0, \\ 1 - a_i & \text{otherwise.} \end{cases} \tag{7}$$

In Step (a), the conversion from the normal basis I_2 to the intermediate basis I_3 can be given as follows. If $A = (a'_0, a'_1, \ldots, a'_{m-1})$ w.r.t. the normal basis I_2, then w.r.t I_3 one has

$$(a'_0, a'_1, \ldots, a'_{m-1}) \mapsto (\underbrace{a'_0, \ldots, a'_0}_{k}, \underbrace{a'_1, \ldots, a'_1}_{k}, \ldots, \underbrace{a'_{m-1}, \ldots, a'_{m-1}}_{k}). \tag{8}$$

The reverse conversion, however, is much more complicated. Note that it is not possible to convert every redundant representation, since some of them may not represent an element in the field $\mathbb{F}_{2^m}$. Two tasks have to be performed in this step: One is to identify the representation of a field element w.r.t. I_3, and the second is to convert the identified field element's representation back to the normal basis.

For the interest of this paper which deals with finite field multiplication, it is sufficient to consider identifying the product of two field elements in I_3 and then convert it back to the normal basis. Suppose that the coordinates $c_i^{(3)}, 1 \leq i \leq n-1$ of the product C w.r.t. I_3 are given. Then, we have the following lemma.

Lemma 1 *Assume that $A, B \in \mathbb{F}_{2^m}$ are respectively given in I_3 by*

$$A = \sum_{j=0}^{m-1} \sum_{i=1}^{k} a_{jk+i}^{(3)} \beta^{2^j \alpha^i} \text{ and } B = \sum_{j=0}^{m-1} \sum_{i=1}^{k} b_{jk+i}^{(3)} \beta^{2^j \alpha^i},$$

where

$$a_{jk+1}^{(3)} = a_{jk+2}^{(3)} = \cdots = a_{jk+k}^{(3)} \text{ and } b_{jk+1}^{(3)} = b_{jk+2}^{(3)} = \cdots = b_{jk+k}^{(3)} \tag{9}$$

for $j = 0, 1, \ldots, m-1$. Then the product $C = AB$ in I_3 obtained using (3) also has the property: $c_{jk+1}^{(3)} = c_{jk+2}^{(3)} = \cdots = c_{jk+k}^{(3)}$ for $j = 0, 1, \ldots, m-1$.

The assertion of the lemma is immediately clear from the following fact. The representations of A and B w.r.t. I_3 are images of a homomorphism ϵ, namely the embedding of $\mathbb{F}_{2^m}$ into $\mathbb{F}_2^{(n)}$. Clearly, $A = \epsilon(a)$ and $B = \epsilon(b)$ for some $a, b \in \mathbb{F}_{2^m}$. Note that the image of the embedding ϵ is characterized by the property expressed in (9). Consequently, $C = A \cdot B = \epsilon(a \cdot b)$ is also in the image of ϵ and thus of the claimed form.

The lemma allows us to identify the I_3 basis representation of the product of two field elements also represented by I_3. Once the product is obtained in this I_3 basis, it can be converted to the corresponding normal basis as

$$(\underbrace{c'_0, \ldots, c'_0}_{k}, \underbrace{c'_1, \ldots, c'_1}_{k}, \ldots, \underbrace{c'_{m-1}, \ldots, c'_{m-1}}_{k}) \mapsto (c'_0, c'_1, \ldots, c'_{m-1}). \tag{10}$$

Thus, Step (a) of basis conversion can be realized with (8) and (10).

3.4 Further Results on Redundant Basis

Lemma 2 *Let $A \in \mathbb{F}_{2^m}$ and consider its I_4 basis representation $(a_1^{(4)}, a_2^{(4)}, \ldots, a_{n-1}^{(4)})$ obtained from its normal basis representation by using (8) and (5). Then the last $\frac{mk}{2}$ coordinates $a_{\frac{n+1}{2}}^{(4)}, a_{\frac{n+3}{2}}^{(4)}, \ldots, a_{n-1}^{(4)}$*

are a mirror reflection of the first $\frac{mk}{2}$ coordinates if $k \geq 2$ is an even integer.

Proof: Let α be a primitive k^{th} root of unity. Then $\gamma = \sum_{i=0}^{k-1} \beta^{\alpha^i}$ generates a normal basis $I_2 = [\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}]$ in $\mathbb{F}_{2^m}$. Since $k \geq 2$ is an even number and $\alpha^{\frac{k}{2}} = -1$, thus

$$\begin{aligned} \gamma^{2^i} &= \beta^{2^i} + \beta^{2^i\alpha} + \beta^{2^i\alpha^2} + \cdots + \beta^{2^i\alpha^{k-1}} \\ &= \beta^{2^i} + \beta^{2^i\alpha} + \cdots + \beta^{2^i\alpha^{\frac{k}{2}-1}} + \beta^{2^i\alpha^{\frac{k}{2}}} + \beta^{2^i\alpha^{\frac{k}{2}+1}} + \cdots + \beta^{2^i\alpha^{k-1}} \\ &= (\beta^{2^i} + \beta^{2^i\alpha} + \cdots + \beta^{2^i\alpha^{\frac{k}{2}-1}}) + (\beta^{-2^i} + \beta^{-2^i\alpha} + \cdots + \beta^{-2^i\alpha^{\frac{k}{2}-1}}). \end{aligned} \tag{11}$$

The two intermediate bases are respectively given by $I_3 = [\beta, \ldots, \beta^{\alpha^{\frac{k}{2}-1}}, \beta^{-1}, \ldots, \beta^{-\alpha^{\frac{k}{2}-1}}, \ldots, \beta^{2^{m-1}}, \ldots, \beta^{2^{m-1}\alpha^{\frac{k}{2}-1}}, \beta^{-2^{m-1}}, \ldots, \beta^{-2^{m-1}\alpha^{\frac{k}{2}-1}}]$ and $I_4 = [\beta, \beta^2, \ldots, \beta^{mk}]$. It can be seen from (11) that the k coordinates of A w.r.t I_4 have the same values. If a line is drawn at the middle of the I_4 basis element sequence between $\beta^{\frac{km}{2}}$ and $\beta^{\frac{km}{2}+1}$, then for any I_4 coordinate $a^{(4)}_{(2^i\alpha^j)}$ its mirror reflection coordinate $a^{(4)}_{n-(2^i\alpha^j)} = a^{(4)}_{(-2^i\alpha^j)}$ must have the same value. □

For example, let $k = 4$ and $m = 7$. Let β denote a primitive 29th root of unity in $\mathbb{F}_{2^{28}}$. Since 12 has order 4 in $\mathbb{Z}^{\times}_{29}$, then $\gamma = \beta + \beta^{12} + \beta^{-1} + \beta^{-12}$ is a Gauß period of type $(7,4)$ over $\mathbb{F}_2$ and γ generates a normal basis in $\mathbb{F}_{2^7}$: $I_2 = [\gamma, \gamma^2, \gamma^4, \ldots, \gamma^{64}]$. Subsequently, I_3 and I_4 are respectively given by $I_3 = [\beta, \beta^{12}, \beta^{28}, \beta^{17}, \beta^2, \beta^{24}, \beta^{27}, \beta^5, \ldots, \beta^6, \beta^{14}, \beta^{23}, \beta^{15}]$ and $I_4 = [\beta, \beta^2, \beta^3, \ldots, \beta^{28}]$. Finally, the redundant representation basis can be obtained by including the element "1" before the element β in I_4. Let the normal basis representation of a field element A be $(a'_0, a'_1, a'_2, a'_3, a'_4, a'_5, a'_6)$. We can obtain its I_4 representation as follows:

$$A = (a'_0, a'_1, a'_5, \underbrace{a'_2, a'_1, a'_6, a'_5, a'_3, a'_3, a'_2, a'_4, a'_0, a'_4, a'_6,}_{\text{8 consecutive coordinates}} a'_6, a'_4, a'_0, a'_4, a'_2, a'_3, a'_3, a'_5, a'_6, a'_1, a'_2, a'_5, a'_1, a'_0).$$

It can be seen that in the I_4 basis representation the first 14 coordinates are a mirror reflection of the last 14 coordinates. The corresponding redundant basis representation of A is obtained simply by including a "0" before the first coordinate in the I_4 representation.

Also note that only eight consecutive coordinates $(a_4, a_5, \ldots, a_{11})$ of the redundant representation, which include all the seven coordinates w.r.t. the normal basis, are necessary for determining the element A. This fact can be exploited when a multiplication operation using

m	n	h	h/n	m	n	h	h/n	m	n	h	h/n
153	613	281	0.46	177	709	326	0.46	207	829	375	0.45
163	653	299	0.46	193	773	357	0.46	213	853	393	0.46
169	677	313	0.46	199	797	368	0.46	219	877	396	0.45
175	701	323	0.46	205	821	374	0.46	235	941	428	0.45

Table 2: Ratios of h to n for some useful values of m with k=4.

redundant representations is implemented. If we denote h as the minimal number of *consecutive* coordinates of the redundant representation needed to determine the element, then Table 2 shows some values of h for the fields given in Table 1 which can be generated with the Gauß period of type $(m, 4)$.

Lemma 3 *For $A, B \in \mathbb{F}_{2^m}$, let $A = (a_0, a_1, \ldots, a_{n-1})$ and $B = (b_0, b_1, \ldots, b_{n-1})$ w.r.t. I_1, where $n = km + 1$. Assume that $C = A \cdot B = (c_0, c_1, \ldots, c_{n-1})$ is obtained using (3). If $a_0 = b_0$ and k is even, then $c_0 = a_0 = b_0$.*

Proof: It follows from Lemma 2, $b_1 = b_{n-1}, b_2 = b_{n-2}, \ldots, b_{\frac{n-1}{2}} = b_{\frac{n+1}{2}}$. Then from (3) we have $c_0 = \sum_{i=0}^{n-1} a_i b_{n-i} = \sum_{i=0}^{n-1} a_i b_i = a_0 b_0 + \sum_{i=1}^{n-1} a_i b_i$. Note that $(a_1, a_2, \ldots, a_{n-1})$ have exactly k copies of the normal basis coordinates, and the same property also applies to $(b_1, b_2, \ldots, b_{n-1})$ (refer to (9)). Then $\sum_{i=1}^{n-1} a_i b_i = \sum_{i=1}^{mk} a_i b_i$ can be written as a sum of m partial sums, where each partial sum is a sum of k same values which is clearly zero since k is even. Then $\sum_{i=1}^{n-1} a_i b_i = 0$ and $c_0 = a_0 b_0$. $\square$

This property will be used later to obtain efficient architecture for finite field multiplier.

4 ARCHITECTURES FOR REDUNDANT BASIS MULTIPLICATION

In this section we present architectures for hardware implementation of the multiplication: $A \cdot B = C$ based on (3), where A, B and C are represented with respect to the redundant basis I_1.

Conversion between I_1 and the normal basis I_2, as discussed in Subsection 3.3, can be performed without any logic gates.

4.1 Bit-Serial Multipliers

Parallel-in serial-out version An architecture for a parallel-in-serial-out (PISO) multiplier is shown in Fig. 1. The n-bit register, which is initially loaded with B, is cyclically shifted with a clock. The contents of this register are bit-wise multiplied with the coordinates of A and the resulting n bits are added using $n-1$ XOR gates (arranged in a binary tree form for minimum delay). For a straightforward implementation, this PISO multiplier requires n flip-flops,[2] n AND gates and $n-1$ XOR gates, and the multiplication is completed in n clock cycles.

The PISO multiplier architecture shown in Fig. 1 can be optimized and its time and space complexities can be greatly reduced if certain properties of the redundant representation basis are taken into consideration. Since the redundant basis coordinates $(a_i)_{i=0}^{n-1}$ contain k copies of the normal basis coordinates a'_j for all $j, (j = 0, 1, \ldots, m-1)$, suppose $a_{i_1} = a_{i_2} = \cdots = a_{i_k}$. If these k coordinates are bit-wise multiplied with $b_{l_1}, b_{l_2}, \ldots, b_{l_k}$, respectively, then part of the PISO multiplier (refer to Fig. 1) computes $a_{i_1}b_{l_1} + a_{i_2}b_{l_2} + \cdots + a_{i_k}b_{l_k}$, which requires k AND and $k-1$ XOR gates. Since $a_{i_1} = a_{i_2} = \cdots = a_{i_k}$, one can equivalently compute $a_{i_1}(b_{l_1} + b_{l_2} + \cdots + b_{l_k})$, which requires only 1 AND and $k-1$ XOR gates. This reduces the total number of AND gates[3] in the PISO multiplier from n to $m+1$, while the number of XOR gates remains the same.

It is also possible to reduce the number of clock cycles needed by the PISO multiplier. Towards this end, if we can change the order of the input bits to the PISO multiplier such that in the first μ ($m \le \mu \le n$) clock cycles the multiplier generates those μ consecutive coordinates of C that have at least one copy of c_j, for all $j, (j = 0, 1, \ldots, m-1)$, then the computation time would reduce from $n(= km + 1)$ to μ clock cycles. The value of μ can be considerably lower than n. Table 2 lists minimum values of μ (denoted as h) for $k = 4$ and $150 \le m \le 250$ that are of interest to elliptic curve cryptosystems. It can be seen from the table that for $k = 4$, the computation time is reduced by more than 50%. In fact for any even value of k, the computation

[2] Note that input A can directly come from a register that is not necessarily part of the multiplier. As a result, in determining the circuit complexity of the multiplier, no register is considered for A.

[3] One more AND gate may be saved if one can ensure that A always has $a_0 = 0$.

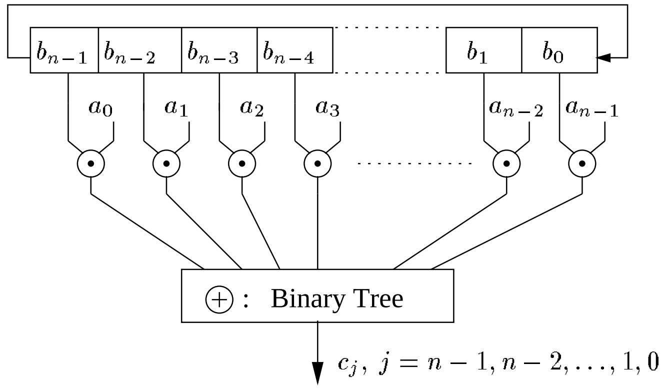

Figure 1: Bit serial multiplier using the redundant representation.

time is always less than $\frac{k}{2}m$. Let the μ consecutive coordinates that the PISO multiplier needs to generate be $c_{l'}, c_{l'+1}, \ldots, c_{l'+\mu-1}$. Then if we change the connection to the AND gates in Fig. 1 such that $(a_0, a_1, \ldots, a_{n-1})$ is replaced by $(a_{l'+\mu}, a_{l'+\mu+1}, \ldots, a_{l'+\mu-1})$, then the PISO multiplier will generate the required μ coordinates of C in the first μ clock cycles.

Serial-in parallel-out version A serial-in parallel-out (SIPO) multiplier which is capable of running at a very high clock rate is shown in Fig. 2, where the element B is stored in a cyclic shift register and the element A is shifted in a bit-serial fashion. Each of the n accumulator units consists of a mod 2 adder and a flip-flop. These flip-flops are initialized to zero and contain the product C after n clock cycles.

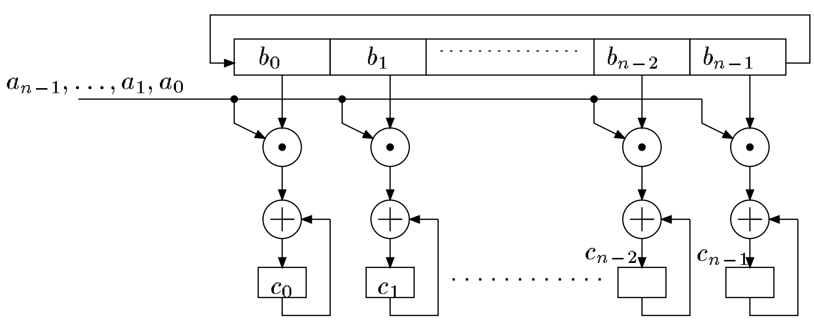

Figure 2: High speed architecture for bit-serial redundant representation multiplier.

Compared to the parallel-in serial-out multiplier of Fig. 1, the SIPO multiplier costs n extra flipflops and one more XOR gate. However, it can support a very high clock rate since the critical path consists of one XOR and one AND gate only. Another clear advantage of this SIPO architecture over the PISO one is that the former can be efficiently implemented in software using the full width of the datapath of the processor on which the software is executed. The optimization for this architecture includes reducing the number of accumulation units to m, such that the results in the m flipflops have exactly one copy of the coordinates w.r.t. to the normal basis.

Table 3 shows a comparison of the two multipliers presented here and the parallel-in serial-out polynomial ring multiplier proposed in [4]. In Table 3, since n' denotes the size of the maximum subfields in cyclotomic rings, for the same field $\mathbb{F}_{2^m}$ one always has $n' \geqslant n$. For example, when $m = 4$, we have $n = n' = 5$; when $m = 5$, we have $n' = 31$ and $n = 11$; when $m = 9$, we have $n' = 73$ and $n = 19$.

Multipliers	# of AND	# of XOR	# Flip flops	# clk cycles	Critical path	basis
Drolet [4]	n'	$n'-1$	n'	n'	$T_A + \lceil \log_2 n' \rceil T_X$	poly. ring
Fig 1(optimized)	m	$n-1$	n	h	$T_A + (\lceil \log_2 k \rceil + \lceil \log_2 m \rceil) T_X$	redundant
Fig 2(optimized)	m	m	$n+m$	n	$T_A + T_X$	redundant

Table 3: Comparison of bit-serial multipliers using polynomial ring basis and redundant representation.

Constant multiplier For an implementation of multiplication operation, if one of the inputs (*i.e.*, either A or B) is known or fixed, the multiplier is called a constant multiplier. In the past, efficient architectures for such constant multipliers were proposed using polynomial and its dual basis. When normal bases are used, the constant multiplier are however not that efficient. This is mainly because most normal basis multipliers require that both A and B be shifted in cyclic fashion in each step of the multiplication operation. To alleviate this problem, the PISO multiplier shown in Fig. 1 can be used with its input A being a fixed element. Although the PISO multiplier's inputs and outputs are represented with respect to the redundant basis, one can change the representation from a normal basis to the redundant basis and vice versa without any logic gates.

When the redundant representation is used, one has another advantage in constructing a constant multiplier. Since a field element can have more than one redundant representation, we may find the representation with the *least* Hamming weight to reduce implementation complexity. For the representations of an element w.r.t. to a redundant basis, the representation with the fewest nonzero coordinates is referred to as the *minimal representation* of the element, and one has the following:

Theorem 2 Let $[1, \beta, \ldots, \beta^{n-1}]$ be a redundant representation basis for $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ and $A \in \mathbb{F}_{2^m}$. Then the minimal representation of A with redundant basis has a Hamming weight equal to or less than $\frac{n-1}{2}$ if $k = 1$ or m is even, and $\frac{n-k+1}{2}$ if $k > 1$ and m is odd.

Proof: The theorem follows by noting that A can be written as

$$\begin{aligned} A &= a_0 + a_1\beta + \cdots + a_{n-1}\beta^{n-1} \\ &= (1 + a_0) + (1 + a_1)\beta + \cdots + (1 + a_{n-1})\beta^{n-1}. \quad \square \end{aligned}$$

4.2 Parallel Architectures

Full parallel version Since the architecture shown in Fig. 1 operates in parallel-in and serial-out fashion, it can be easily parallelized. Fig. 3 shows the circuit (module M) that generates one coefficient c_i of the product C. The inputs to module M are A and i-fold cyclically shifted version of B. Clearly, a full bit-parallel multiplier can be obtained by using n such modules M.

The circuits for module M can be optimized to save AND gates in the same way as we discussed for the PISO multiplier. Also the number of modules can be reduced to m. Since it is sufficient to generate only those m coordinates that correspond to the normal basis, each module M requires m AND and $n-1$ XOR gates, and there are m such modules M.[4] Hence the total number of gates for the bit parallel multiplier is

$$\begin{array}{rl} m^2 & \text{AND gates,} \\ m(n-1) & \text{XOR gates.} \end{array}$$

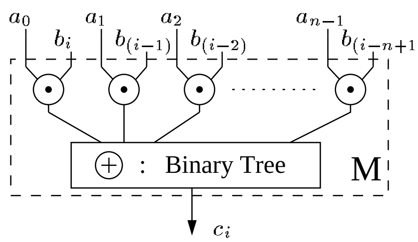

Figure 3: Parallelization of the bit-serial redundant representation multiplier.

The time delay due to gates is $T_A + (\lceil \log_2 k \rceil + \lceil \log_2 m \rceil) T_X$.

Hybrid version The above bit parallel architecture has a clear advantage over some similar existing architectures. It can be implemented in partial parallel (hybrid) fashion to provide considerable amount of space and time trade-offs. In a space constrained environment, if only t copies of modules M are available to implement the multiplier ($1 \leq t \leq n$), then the multiplication operation can be arranged such that in one clock cycle t c_j's are computed, and the operation can be completed in $\left\lceil \frac{n}{t} \right\rceil$ clock cycles. This feature could be very useful in VLSI implementation since it might be difficult to implement a full-scale bit-parallel multiplier when the field is very large.

Fig. 4 shows the architecture of a hybrid multiplier using only two modules M. There are two shift registers R_1 and R_2. Register R_1 is of length $\frac{n+1}{2}$ bits and initially loaded with $b_1, b_3, \ldots, b_{n-2}, b_0$. Register R_2 is $\frac{n-1}{2}$ bits long and initially loaded with $b_2, b_4, \ldots, b_{n-1}$. The *interlacing* module combines the outputs from the two registers into one such that its first bit is the first bit from R_1, the second bit is the first bit from R_2, the 3rd bit is the second bit from R_1, the 4th bit is the second bit from R_2, …, and so on. During the first clock cycle, the interlacing module has outputs in the order: $b_1, b_2, \ldots, b_{n-1}, b_0$. Then module M on the left-hand side generates c_0 and module M on the right-hand side produces c_1. During the second clock cycle, the outputs of the interlacing module is $b_3, b_4, \ldots, b_{n-1}, b_0, b_1, b_2$ and c_2 and c_3 are generated by the modules M. This process is repeated and after a total of $\left\lceil \frac{n}{2} \right\rceil$ clock cycles, all the coordinates of C are generated.

[4]Here we assume that k is even and the first coordinates of A and B, $a_0 = b_0 = 0$. Then we have $c_0 = 0$ by Lemma 3. If the above condition is not satisfied, there should be $m + 1$ modules.

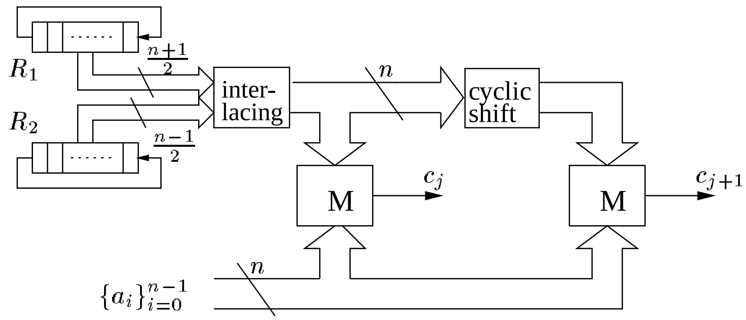

Figure 4: Hybrid redundant representation multiplier architecture ($t = 2$).

5 ARCHITECTURE FOR TYPE-II ONB MULTIPLIER

In this section we deal with type-II ONB. Extending the work of Gao and Vanstone [6], we present several bit-serial and bit-parallel multiplier architectures.

5.1 Algorithm

Below we consider in more detail Remark 2 given in Section 2.

Theorem 3 [6] Let β be a primitive $(2m+1)^{\text{st}}$ root of unity in $\mathbb{F}_{2^m}$ and $\gamma = \beta + \frac{1}{\beta}$ generates a type II optimal normal basis. Then $\{\gamma_i, i = 1, 2, \ldots, m\}$ with $\gamma_i = \beta^i + \frac{1}{\beta^i} = \beta^i + \beta^{2m+1-i}$, $i = 1, 2, \ldots, m$, is also a basis in $\mathbb{F}_{2^m}$.

From the discussion in the previous section, the complexities of a redundant basis multiplier can be greatly reduced by applying certain properties of the redundant representations. However, we can do better. For $B \in \mathbb{F}_{2^m}$ and γ_i as defined in Theorem 3, define

$$s(i) = \begin{cases} i \bmod 2m+1, & \text{if } 0 \leqslant i \bmod 2m+1 \leqslant m, \\ 2m+1-i \bmod 2m+1, & \text{otherwise.} \end{cases} \tag{12}$$

Obviously, $s(0) = 0, s(i) = s(2m+1-i)$ and $\gamma_i = \gamma_{s(i)}$ for any integer i. As $\gamma_i\gamma_j = \gamma_{i+j} + \gamma_{i-j}$, we have $\gamma_i \cdot \gamma_j = \gamma_{s(i+j)} + \gamma_{s(i-j)}$. Let $B = (b_1, \ldots, b_m) \in \mathbb{F}_{2^m}$ with respect to the basis

$[\gamma_1, \gamma_2, \ldots, \gamma_m]$ and $b_0 = 0$, then

$$\gamma_i \cdot B = \sum_{j=1}^{m} b_j \gamma_i \cdot \gamma_j = \sum_{j=1}^{m} b_j(\gamma_{s(i+j)} + \gamma_{s(i-j)}) = \sum_{j=1}^{m} (b_{s(j+i)} + b_{s(j-i)})\gamma_j.$$

The final step in the above equation comes from proper substitutions of the subscript variables. The above constant multiplication $\gamma_i \cdot B$ was proposed by Gao and Vanstone [6]. In order to obtain a general multiplier, let $A = (a_1, \ldots, a_m)$ be an element in $\mathbb{F}_{2^m}$, w.r.t. the basis $[\gamma_1, \gamma_2, \ldots, \gamma_m]$, then multiplication of A and B can proceed as follows:

$$A \cdot B = \sum_{i=1}^{m} a_i(\gamma_i \cdot B) = \sum_{i=1}^{m} a_i \sum_{j=1}^{m} (b_{s(j+i)} + b_{s(j-i)})\gamma_j = \sum_{j=1}^{m} \Big(\sum_{i=1}^{m} a_i(b_{s(j+i)} + b_{s(j-i)}) \Big) \gamma_j.$$

If the product is written as $C = \sum_{j=1}^{m} c_j \gamma_j$, then

$$c_j = \sum_{i=1}^{m} a_i(b_{s(j+i)} + b_{s(j-i)}), \quad j = 1, 2, \ldots, m. \tag{13}$$

Note that γ also generates a normal basis $[\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}] = [\gamma_1, \gamma_2, \ldots, \gamma_{2^{m-1}}]$. From $\gamma_i = \gamma_{s(i)}$ and the expression (12), it can be seen that the basis $[\gamma_1, \gamma_2, \ldots, \gamma_m]$ is a permutation of the above normal basis. Thus in hardware a squaring operation using the basis $[\gamma_1, \gamma_2, \ldots, \gamma_m]$ costs nothing but rearrangement of wires.

5.2 Architectures

Parallel-in serial-out multiplier An architecture to implement this multiplication is shown in Figure 5. A $(2m+1)$-bit register, which is divided into two parts (left and right) and is shifted cyclically, stores $b_{s(i)}, i = 0, 1, \ldots, 2m$. A total of m AND gates and m XOR gates are used to generate m terms of $a_i b_j$'s. Finally, another $m - 1$ XOR gates, formed as a binary tree, take m terms of $a_i b_j$'s as inputs and produce the coordinate c_j of C. In one clock cycle, the register is shifted once and one c_j is generated at the output port. A multiplication is completed in m clock cycles.

The size complexity of the multiplier in Fig. 5 is m AND gates and $2m - 1$ XOR gates, along with a $(2m+1)$-bit shift register. The delay in the critical path is $T_A + (1 + \lceil \log_2 m \rceil) T_X$. A comparison of the proposed multiplier with some other similar bit-serial normal basis multipliers is shown in Table 4. As it can be seen except for the multiplier of Geiselmann and Gollmann [9], the proposed multiplier has an overall space and time complexities that is better than those of any other multiplier listed in the table. The multiplier of [9] requires about $\frac{m}{2}$ fewer XOR gates, however, the proposed multiplier has a highly regular structure which makes it attractive for hardware implementation for very large fields.

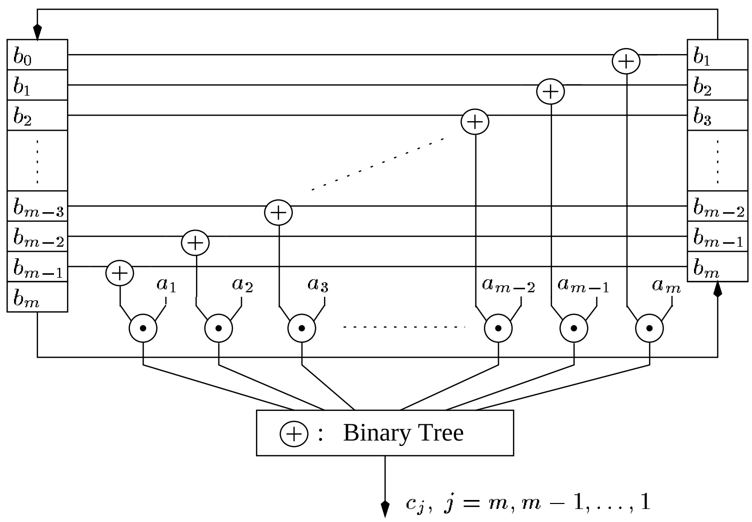

Figure 5: New bit-serial multiplier using basis $[\gamma_i]$.

Multipliers	#AND	#XOR	#flipflops	# clk cycles	basis
Massey-Omura [16]	$2m-1$	$2m-2$	$2m$	m	normal
Feng [5]	$2m-1$	$3m-2$	$3m-2$	m	normal
Agnew *et al* [2]	m	$2m-1$	$3m$	m	normal
Geiselmann-Gollmann[9]	m	$\frac{3m-1}{2}$	$2m$	m	normal
presented here	m	$2m-1$	$2m+1$	m	normal [5]

[5]In fact, it is a permutation of the normal basis.

Table 4: Comparison of bit-serial multipliers when there is a type II ONB.

Serial-in parallel-out multiplier Similar to the redundant basis multiplier architecture discussed in Section 4, a high speed architecture for the modified redundant basis multiplier is also available, which is shown in Fig. 6. The coefficients $b_1, b_1, \ldots, b_m$ of the element B, w.r.t. the basis $[\gamma_i]$, are initially stored in a register of length $2m+1$ which can be shifted cyclically. The coefficients $a_m, a_{m-1}, \ldots, a_1$ of the element A, w.r.t the basis $[\gamma_i]$, are fed into the system from the left in a bit-serial fashion. There are m accumulation units and they are the same as those in Fig. 2. During the first clock cycle, the data $b_0+b_1, b_1+b_2, \ldots, b_{m-1}+b_m$ are respectively multiplied with a_m at the m bit-multipliers. Note that $b_0 = 0$. The m bit-products, which are $a_m(b_0+b_1), a_m(b_1+b_2), \ldots, a_m(b_{m-2}+b_{m-1}), a_m(b_{m-1}+b_m)$, are then stored in the m accumulation units. After the second clock cycle, m values of $a_m(b_0+b_1)+a_{m-1}(b_1+b_2)$, $a_m(b_1+b_2)+a_{m-1}(b_0+b_3), \ldots$, $a_m(b_{m-2}+b_{m-1})+a_{m-1}(b_{m-3}+b_m)$, $a_m(b_{m-1}+b_m)+a_{m-1}(b_{m-2}+b_m)$ are respectively stored in m flipflops. After m clock cycles, the contents of the m flipflops at the top are the coordinates of the product C.

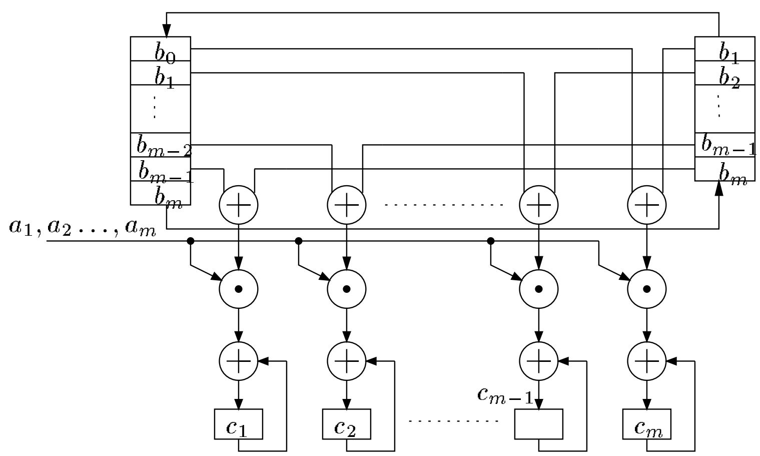

Figure 6: High speed multiplier architecture for a type II ONB.

Compared to the multiplier shown in Fig. 5, the high-speed version multiplier has a higher complexity. Besides m AND gates, $2m$ XOR gates, and a cyclic shift register of length $2m+1$, the high-speed version multiplier also needs m flipflops. The critical path has a delay of T_A+2T_X, which is however much shorter than that of the multiplier shown in Fig. 5.

Hybrid multiplier architecture The bit-serial multiplier shown in Fig. 5 can be easily made parallel or partial parallel. Fig. 7 shows an architecture of a hybrid multiplier using the basis $[\gamma_i]$, which yields two out of m coordinates of the product per clock cycle, and completes a

Multipliers	#AND	#XOR	# clk cycles $\times$ cycle period
Massey-Omura [12]	$(2m-1)t$	$(2m-2)t$	$\left\lceil \frac{m}{t} \right\rceil \times [T_A + \lceil \log_2(2m-1) \rceil T_X]$
Proposed here	mt	$(2m-1)t$	$\left\lceil \frac{m}{t} \right\rceil \times [T_A + (1 + \lceil \log_2 m \rceil) T_X]$

Table 5: Comparison of hybrid multipliers when there is a type II ONB ($1 \leqslant t \leqslant m+1$).

multiplication operation in $\left\lceil \frac{m}{2} \right\rceil$ clock cycles. Note that bit-serial and full parallel multipliers can be viewed as special cases of the hybrid architecture.

It can be seen from Fig. 7 that module M' is all combinatorial circuits and similar to module M in Fig. 3. In Fig. 7, two copies of module M' are used and each of them generates one product coordinate at a time. The cyclic shift module enables a cyclic shift of $2m+1$ coefficients $b_{s(0)}, b_{s(1)}, \ldots, b_{s(2m)}$, and costs no gates and registers. When m is even, the $2m+1$-bit register is initially loaded with $b_0, b_2, \ldots, b_m$,$b_{m-1}, b_{m-3}, \ldots, b_1$,$b_1, b_3, \ldots, b_{m-1}, b_m$,$b_{m-2}, \ldots, b_2$. When m is an odd number, the order of the $2m+1$ bits initially loaded into the register is $b_0, b_2, \ldots, b_{m-1}, b_m$,$b_{m-2}, \ldots, b_1$,$b_1, b_3, \ldots, b_m$,$b_{m-1}, b_{m-3}, \ldots, b_2$. The permutation module takes the $2m+1$ bits from the shift register and during the first clock cycle its output is in the order of $b_{s(0)}, b_{s(1)}, \ldots, b_{s(2m)}$. Values of $s(i)$ can be calculated using (12) and always lie between 0 and m, inclusive, for $i = 0, 1, \ldots, 2m$. Note that the module M', which generates c_j, takes $2m$ out of $2m+1$ bits from $b_{s(0)}, \ldots, b_{s(2m)}$ and leaves the bit $b_{s(j)}$ out.

Obviously, the multiplier's space complexity depends on how many modules M' are used in the partially parallel architecture. Each module M' consists of $2m-1$ XOR gates and m AND gates, which is shown in the right-hand side in Fig. 7. The total complexity of the hybrid multiplier with two modules M' is $4m-2$ XOR gates and $2m$ AND gates. Comparison between this work and the bit-parallel Massey-Omura multiplier proposed by Wang et al. [20] is made and shown in Table 5. It can be seen that with the same number of XOR gates used and approximately same time delay, the multiplier presented here uses about only half the number of AND gates used in the Massey-Omura multiplier.

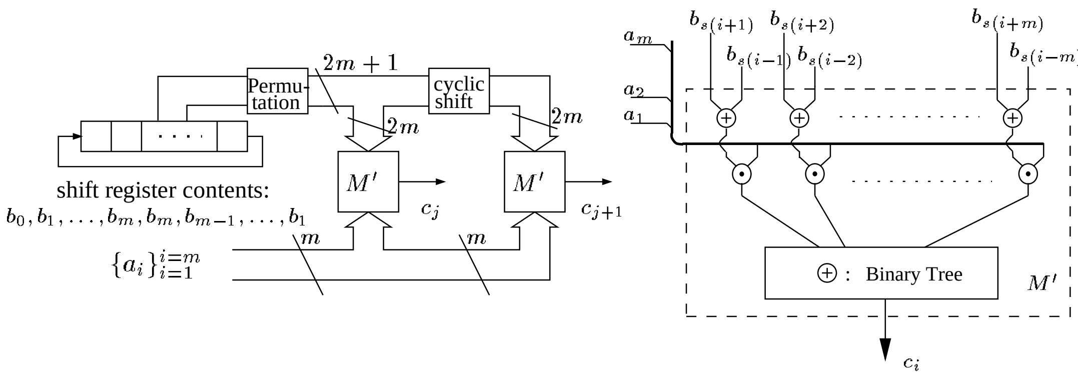

Figure 7: Bit-parallel multiplier using basis $[\gamma_i]$.

6 CONCLUDING REMARKS

In this paper, we have considered multiplication in $\mathbb{F}_{2^m}$ using a redundant representation. The basic idea behind the multiplication is to embed the field $\mathbb{F}_{2^m}$ into the smallest cyclotomic field $\mathbb{F}_2^{(n)}$ and do the arithmetic in $\mathbb{F}_2^{(n)}$. We have presented the smallest n for various values of m that are of practical interest for elliptic curve cryptosystem.

We have also shown that the redundant representation can be used to obtain efficient bit-serial, bit-parallel, and hybrid multiplier structures. Additionally. we have discussed how to reduce the time and space complexities of these multipliers using properties of the redundant representation.

The conversions from the redundant representation to the corresponding normal basis and vice versa have been given. We have shown that these conversions can be implemented in hardware without any logic gates.

When there is a type I ONB in $\mathbb{F}_{2^m}$, it follows from our discussion in Section 4 that the minimal representation of a constant field element always has a Hamming weight not greater than $\frac{m}{2}$. Consequently, the proposed constant multiplier has very low complexity. When there exists a type II ONB, very simple and highly regular multiplier architecture can be obtained using the redundant representation (refer to Section 5). It has been shown that such multipliers have lower or equivalent complexity compared to most of the previously proposed similar multipliers. Hybrid or partial parallel architectures have also been presented for this type of ONBs.

One question arising from the work presented here remains: Can this modified redundant representation multiplier described in Section 5 be generalized to any field $\mathbb{F}_{2^m}$?

Acknowledgment

Part of this manuscript was presented at the *Workshop on Cryptographic Hardware and Embedded Systems'99*, August, 1999, Worcester, MA [23]. We thank the anonymous reviewers for their useful and valuable comments. Dr. Wu also thanks Ellen Liu for her helpful discussions during the preparation of the manuscript.

References

[1] G. B. Agnew, R. Beth, R. C. Mullin, and S. A. Vanstone. Arithmetic operations in GF(2^m). *J. Cryptology*, 6:3–13, 1993.

[2] G. B. Agnew, R. C. Mullin, I. Onyszchuk, and S. A. Vanstone. An implementation for a fast public key cryptosystem. *J. Cryptology*, 3:63–79, 1991.

[3] D. W. Ash, I. F. Blake, and S. A. Vanstone. Low complexity normal bases. *Disc. Appl. Math.*, 25:191–210, 1989.

[4] G. Drolet. A new representation of elements of finite fields GF(2^m) yielding small complexity arithmetic circuits. *IEEE Trans. Comput.*, 47(9):938–946, 1998.

[5] M. Feng. A VLSI architecture for fast inversion in GF(2^m). *IEEE Trans. Comput.*, 38:1383–1386, 1989.

[6] S. Gao and S. Vanstone. On orders of optimal normal basis generators. *Math. Comp.*, 64(2):1227–1233, 1995.

[7] S. Gao, J. von zur Gathen, and D. Panario. Gauss periods and fast exponentiation in finite fields. In *Lecture Notes in Computer Science*, volume 911, pages 311–322. Springer-Verlag, 1995.

[8] S. Gao, J. von zur Gathen, D. Panario, and V. Shoup. Algorithms for exponentiation in finite fields. *J. of Symbolic Computation*, 29:879–889, 2000.

[9] W. Geiselmann and D. Gollmann. VLSI design for exponentiation in GF(2^m). In *AUSCRYPT'90*, pages 398–405. Springer-Verlag, 1990.

[10] W. Geiselmann and H. Lukhaub. Redundant representation of finite fields. In *Proceedings of Public Key Cryptography*, pages 339–352. Springer-Verlag, 2001.

[11] 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. Comput.*, 41(8):962–971, 1992.

[12] M. A. Hasan, M. Wang, and V. K. Bhargava. A modified Massey-Omura parallel multiplier for a class of finite fields. *IEEE Trans. Comput.*, 42(8):1278–1280, 1993.

[13] T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverse in GF(2^m) using normal bases. *Inform. and Comput.*, 78:171–177, 1988.

[14] T. Itoh and S. Tsujii. Structure of parallel multipliers for a class of fields GF(2^m). *Inform. and Comput.*, 83:21–40, 1989.

[15] R. Lidl and H. Niederreiter. *Finite Fields*. Addison-Wesley Publishing Company, Reading, MA, 1983.

[16] J. L. Massey and J. K. Omura. Computational method and apparatus for finite field arithmetic. U.S. Patent No.4587627, 1984.

[17] R. Mullin, I. Onyszchuk, S. A. Vanstone, and R. Wilson. Optimal normal bases in GF(p^n). *Disc. Appl. Math.*, 22:149–161, 1988.

[18] I.M. Onyszchuk, R.C. Mullin, and S.A. Vanstone. Computational method and apparatus for finite field multiplication. U.S. Patent No.4,745,568, 1988.

[19] J. H. Silverman. Fast multiplication in finite field GF(2^N). In Ç. K. Koç and C. Paar, editors, *CHES'99, LNCS 1717*, pages 122–134. Springer-Verlag, 1999.

[20] 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. Comput.*, 34(8):709–717, 1985.

[21] A. Wassermann. Konstruktion von Normalbasen. *Bayreuther Mathematische Schriften*, pages 155–164, 1990.

[22] J. K. Wolf. Low complexity finite field multiplication. *Disc. Math.*, 106/107:497–502, 1992.

[23] H. Wu, M. A. Hasan, and I. F. Blake. Highly regular architectures for finite field computation using redundant basis. In Ç. K. Koç and C. Paar, editors, *CHES'99, LNCS 1717*, pages 269–279. Springer-Verlag, 1999.