# Revisiting Finite Field Multiplication Using Dickson Bases

Bijan Ansari and M. Anwar Hasan

Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada
{bansari, ahasan}@uwaterloo.ca

**Abstract.** Dickson bases have recently been introduced in [1] for finite field arithmetic. Such a basis exists for any extension field and under certain conditions it represents a permutation of type II optimal normal bases. In this paper Dickson bases are developed using simpler mathematical terms and their properties are discussed. An algorithm/architecture based on the model presented in [1] is developed. An alternate simpler multiplication algorithm is introduced, the results are compared to a typical serial polynomial base multiplier and VLSI implementation considerations are discussed.

Keywords: Dickson bases, Dickson polynomial, finite fields, field multiplication.

## 1 Introduction

Practical applications of finite fields include channel coding, VLSI testing and cryptography. An extended finite field can be viewed as a vector space over a ground field. A set of linearly independent vectors which span the vector space is called a basis and can be used to represent the elements of the finite extension field. Selection of a basis is often influenced by the target application. For efficient field arithmetic, bases that are commonly used include polynomial, normal, dual and triangular bases.

Dickson bases have recently been introduced in [1] and it is proved that such a basis exists for any finite field extension. A computation model for multiplication over $\mathbb{F}_{2^n}$ along with its optimization for a fixed reduction polynomial is also provided to show the feasibility of implementation. In this paper we develop Dickson bases and explain their properties using simpler mathematical terms that are more suitable to practitioners. Then we present a multiplication algorithm based on the model presented in [1] and discuss VLSI implementation considerations. Finally, we present a new multiplication algorithm using Dickson bases which is simpler than the model presented in [1] and compare Dickson bases multiplication algorithms with polynomial basis multiplication for binary extension field $\mathbb{F}_{2^n}$.

The following notations are used in this paper:
$p$ : a prime number
$q$ : a power of prime $p$, i.e., $p^n$ where integer $n > 1$
$A$ : a register which holds an element of $\mathbb{F}_{p^n}$
$\mathbf{a}$ : an element of $\mathbb{F}_{p^n}$
$a_i$ : the $i^{\text{th}}$ coordinate of $\mathbf{a}$, where $\mathbf{a} \in \mathbb{F}_{p^n}$ and $a_i \in \mathbb{F}_p$
$(a_n, \ldots, a_1, a_0)$: coordinates of $\mathbf{a} \in \mathbb{F}_{p^n}$
$A[i \ldots j]$: elements $i$ to $j$ of register $A$
$A[i]$: the $i^{\text{th}}$ element of register $A$
$\mathbf{r}$ : an irreducible polynomial whose coefficients are over $\mathbb{F}_p$

## 2 Motivation and Mathematical Background

An extended finite field of $p^n$ elements is denoted as $\mathbb{F}_{p^n}$, where $p$ is prime and $n$ is an integer greater than one. Elements of $\mathbb{F}_{p^n}$ are represented in terms of basis $\{b_{n-1}, \ldots, b_1, b_0\}$ over the ground field $\mathbb{F}_p$. The most common basis used in finite field arithmetic is called the polynomial or canonical basis and it has the following form $\{\alpha^{n-1}, \ldots, \alpha^1, \alpha^0\}$ , where $\alpha \in \mathbb{F}_{p^n}$. For field arithmetic, the other bases that have been shown to be suitable for practical applications include normal, dual and triangular bases. There are two types of normal bases which allow efficient implementation of finite field multiplication. They are optimal normal bases (ONB) of type I and type II and are only available for certain $n$ and $p$ [2]. Finite field arithmetic using a Dickson basis has recently been presented in [1]. It is shown that set $\mathbf{B} = \{\beta_{n-1}, \ldots, \beta_2, \beta_1, 1\}$, where $\beta_i$'s are Dickson polynomials $D_i(\beta, 1)$ of the first kind over $\mathbb{F}_p$ and 1 is the multiplicative identity element, is a basis for $\mathbb{F}_{p^n}$. Under certain conditions Dickson polynomials represent a permutation of the type II optimal normal basis. For finite fields where optimal normal bases do not exist, the use of Dickson bases is simply an attempt to efficiently perform field arithmetic, in particular multiplication.

### 2.1 Dickson Polynomials

Dickson bases can be developed without a direct reference to Dickson polynomials. Here, we briefly explain some Dickson polynomial properties because they give a better understanding to the computation and implementation of finite field arithmetic using Dickson bases. The reader is refereed to [3] for more details on Dickson polynomials.

Dickson polynomials over finite fields were introduced by L.E.Dickson [4]. There are a number of ways to define Dickson polynomials. Here, we use the recursive definition. Dickson polynomial over ring $R$ of the first kind of degree $n$ in the indeterminate $x$ with parameter $a \in R$ can be defined using the recursive relation

$$D_{n+1}(x,a) = xD_n(x,a) - aD_{n-1}(x,a) \quad n \geq 0, \tag{1}$$

with the following initial conditions

$$D_0(x,a) = 2, \quad D_1(x,a) = x.$$

If we set $x = u + au^{-1}$, then by using mathematical induction and (1) it can be shown that

$$D_n(u + \frac{a}{u}, a) = u^n + \left(\frac{a}{u}\right)^n. \tag{2}$$

Using (2) and setting $x = u + au^{-1}$, it is readily verified that

$$D_i(x,1)D_j(x,1) = D_{i+j}(x,1) - D_{|i-j|}(x,1). \tag{3}$$

Changing the indeterminate $x$ to $\beta$ and renaming $D_i(\beta,1)$ to $\beta_i$ (3) can be written as

$$\beta_i\beta_j = \beta_{i+j} + \beta_{|i-j|}. \tag{4}$$

Setting $j$ to 1 in (4) and noting that $\beta_1 = \beta$, for $i > 1$ we obtain

$$\beta_{i+1} = \beta_1\beta_i - \beta_{i-1}, \tag{5}$$

which can be obtained from (1) as well. This work uses (4) and (5) extensively. Table 1 is produced using (5) and shows Dickson polynomials of degree from one to nine with indeterminate $\beta$ and parameter $a = 1$ over $\mathbb{Z}$ and $\mathbb{F}_2$.

**Table 1.** Dickson polynomials degree zero to nine over $\mathbb{Z}$ and $\mathbb{F}_2$

| $i$ | $\beta_i$ | $D_i(\beta,1)$ over $\mathbb{Z}$ | $D_i(\beta,1)$ over $\mathbb{F}_2$ |
|---|---|---|---|
| 0 | $\beta_0$ | $2$ | $0$ |
| 1 | $\beta_1$ | $\beta$ | $\beta$ |
| 2 | $\beta_2$ | $\beta^2 - 2$ | $\beta^2$ |
| 3 | $\beta_3$ | $\beta^3 - 3\beta$ | $\beta^3 + \beta$ |
| 4 | $\beta_4$ | $\beta^4 - 4\beta^2 + 2$ | $\beta^4$ |
| 5 | $\beta_5$ | $\beta^5 - 5\beta^3 + 5\beta$ | $\beta^5 + \beta^3 + \beta$ |
| 6 | $\beta_6$ | $\beta^6 - 6\beta^4 + 9\beta^2 - 2$ | $\beta^6 + \beta^2$ |
| 7 | $\beta_7$ | $\beta^7 - 7\beta^5 + 14\beta^3 - 7\beta$ | $\beta^7 + \beta^5 + \beta$ |
| 8 | $\beta_8$ | $\beta^8 - 8\beta^6 + 20\beta^4 - 16\beta^2 + 2$ | $\beta^8$ |
| 9 | $\beta_9$ | $\beta^9 - 9\beta^7 + 27\beta^5 - 30\beta^3 + 9\beta$ | $\beta^9 + \beta^7 + \beta^5 + \beta$ |

## 2.2 Dickson Algebra

Let $\mathbf{S} = \{\ldots, \beta_i, \ldots, \beta_2, \beta_1, 1\}$ be the set of all Dickson polynomials over $\mathbb{F}_p$ and $\mathbf{D}$ be the set of polynomials represented in terms of elements of $\mathbf{S}$, i.e. $\mathbf{D} = \{a_m\beta_m + \ldots + a_1\beta_1 + a_0 | m \in \mathbb{N}, a_i \in \mathbb{F}_p, \beta_i \in \mathbf{S}\}$. We show that $\mathbf{D}$ and $\mathbb{F}_p[x]$ are isomorphic. In other words, $\mathbf{D}$ and $\mathbb{F}_p[x]$ are the same set.

Assume $a(\beta) = \sum_{i=0}^{m} a_i\beta_i \in \mathbf{D}, a_m \neq 0$. Set $\beta_0 = 2$ and $\beta_1 = \beta$. Using (5) it follows that $\beta_i$ is a polynomial of degree $i$ over $\mathbb{F}_p$. Therefore, $a(\beta)$ is a polynomial of degree $m$ over $\mathbb{F}_p$, represented in terms of indeterminate $\beta$. Conversely, assume that $a'(x) = \sum_{i=0}^{m} a_i x^i \in \mathbb{F}_p[x]$. Using $x = \beta = \beta_1$ and $\beta^n = \beta_1\beta^{n-1}$, any $x^i$ can be expressed in terms of elements of $\mathbf{S}$; hence $a'(x) \in \mathbf{D}$. Therefore $\mathbf{D}$ can be viewed as a set of polynomials over $\mathbb{F}_p$, represented by the elements of $\mathbf{S}$. Addition and multiplication can be performed over $\mathbf{D}$. Addition is easily performed by adding the coefficients of each $\beta_i$. The multiplication of two Dickson polynomials, namely $\beta_i\beta_j$, can be expressed in terms of addition of elements of $\mathbf{S}$ by applying (3). Hence, multiplication can be performed as shown in the following example.

*Example 1.* Assume that $\mathbf{a}, \mathbf{b} \in \mathbf{D}$ are defined over $\mathbb{F}_p$ and represented in terms of $\beta_i$ as $\mathbf{a} = a_3\beta_3 + a_2\beta_2 + a_1\beta_1 + a_0$ and $\mathbf{b} = b_3\beta_3 + b_2\beta_2 + b_1\beta_1 + b_0$. To calculate $\mathbf{ab}$ we need to find the product of the $\beta_i$'s. Using (4) we get $\beta_1\beta_1 = \beta_2 + \beta_0$, $\beta_1\beta_2 = \beta_3 + \beta_1$, $\beta_1\beta_3 = \beta_4 + \beta_2$, $\beta_2\beta_2 = \beta_4 + \beta_0$, $\beta_2\beta_3 = \beta_5 + \beta_1$, $\beta_3\beta_3 = \beta_6 + \beta_0$ and $\beta_0 = 2$. The partial product $(a_1\beta_1)(b_2\beta_2)$ equals to $(a_1b_2)(\beta_1\beta_2) = a_1b_2\beta_3 + a_1b_2\beta_1$. Extending this idea, the entire product of $\mathbf{a}$ and $\mathbf{b}$ is computed as follows:

| $\beta_6$ | $\beta_5$ | $\beta_4$ | $\beta_3$ | $\beta_2$ | $\beta_1$ | $1$ |
|---|---|---|---|---|---|---|
| | | | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
| | | $\times$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
| | | | $a_0b_3$ | $a_0b_2$ | $a_0b_1$ | $a_0b_0$ |
| | | $a_1b_3$ | $a_1b_2$ | $a_1b_1$ | $a_1b_0$ | |
| | | | | $a_1b_3$ | $a_1b_2$ | $\beta_0a_1b_1$ |
| | $a_2b_3$ | $a_2b_2$ | $a_2b_1$ | $a_2b_0$ | | |
| | | | | | $a_2b_1$ | |
| | | | | | $a_2b_3$ | $\beta_0a_2b_2$ |
| $a_3b_3$ | $a_3b_2$ | $a_3b_1$ | $a_3b_0$ | | | |
| | | | | $a_3b_1$ | $a_3b_2$ | $\beta_0a_3b_3$ |

The sum of the columns gives the final product. *Note that $\beta_0$ is not a basis element and it is equivalent to zero when* $\mathbf{a}, \mathbf{b}$ *are defined over fields of characteristic two.*

## 2.3 Dickson Bases

Similar to any other polynomial, irreducible polynomials can be represented in terms of Dickson polynomials as well.

*Example 2.* Consider the polynomial $\mathbf{r} \in \mathbf{D}$ defined over $\mathbb{F}_2$ and let $\mathbf{r} = \beta_3 + 1$. It is an irreducible polynomial over $\mathbb{F}_2$, which can be used to construct $\mathbb{F}_{2^3}$. This field is the same as the field generated by the irreducible polynomial $f(x) = x^3 + x + 1$ over $\mathbb{F}_2$, since $\mathbf{r} = \beta_3 + 1 = \beta^3 + \beta + 1$. The Dickson basis is $\mathbf{B}_3 = \{\beta_2, \beta_1, 1\}$ or $\mathbf{B}_3 = \{\beta^2, \beta, 1\}$ if we replace $\beta_1$ and $\beta_2$ from Table 1.

As has been stated earlier, Dickson polynomial can be used as a basis to represent a finite field. If $\mathbf{r} \in \mathbf{D}$ is an irreducible polynomial of degree $n$ and $\mathbf{D}$ is defined over $p$, then $\mathbf{K} = \mathbf{D}/(\mathbf{r})$ is a finite field of dimension $n$ over $\mathbb{F}_p$ i.e. $\mathbf{K} = \mathbb{F}_{p^n}$ and the set $\mathbf{B} = \{\beta_{n-1}, \ldots, \beta_2, \beta_1, 1\}$ is a basis [1]. To verify that $\mathbf{B}$ is a basis we note that elements of $\mathbf{B}$ are all linear combination of the polynomial basis $\{\beta^{n-1}, \ldots, \beta^2, \beta, 1\}$.

Similar to polynomial bases, irreducible polynomials with a lower number of non-zero terms lead to less complexity in reduction operation using Dickson bases. It is shown in [1], binomial and monomial irreducibles can be found in $\mathbf{D}$.

## 3 Multiplication

In this section, an algorithm for multiplication of two elements $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{p^n}$, where $\mathbf{a}, \mathbf{b}$ are represented using a Dickson basis, is presented. The algorithm performs multiplication in $n-1$ clock cycle and reduces the result *mod* an irreducible polynomial $\mathbf{r}$ in another $n-1$ cycles. In the next section, we will present another algorithm that uses the recursive relation of the basis to perform the multiplication and reduction in $n$ clock cycles.

Let $\mathbf{B} = \{\beta_{n-1}, \ldots, \beta_1, 1\}$ be a Dickson basis. Thus $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{p^n}$ are represented in the Dickson basis as

$$\mathbf{a} = a_0 + \sum_{i=1}^{n-1} a_i \beta_i, \; \mathbf{b} = b_0 + \sum_{i=1}^{n-1} b_i \beta_i$$

and, for short, by their coordinates $(a_{n-1}, \ldots, a_1, a_0)$ and $(b_{n-1}, \ldots, b_1, b_0)$ where $a_i, b_i \in \mathbb{F}_p$. Let

$$\mathbf{r} = \beta_n + \sum_{i=1}^{n-1} r_i \beta_i + r_0 \qquad (6)$$

be an irreducible polynomial over $\mathbb{F}_p$. Our goal is to find the coordinates of $\mathbf{c} = \mathbf{ab}$ (*mod* $\mathbf{r}$), where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}_{p^n} \cong \mathbf{D}/(\mathbf{r})$. Since $\beta_0$ is not a basis element, $a_0$ and $b_0$ need special treatment in the multiplication algorithm.

## 3.1 Two-Pass Multiplication Algorithm

In [1], a computation model is presented where a field multiplication is performed in two passes: multiplication and reduction. Below we give a development of the model/algorithm and map it on a hardware architecture that is regular and can be suitable for implementation.

**Step 1– Multiplication** The multiplication algorithm is similar to the left-to-right serial (bit-serial in $\mathbb{F}_{2^n}$) multiplication in an algebraic field. Multiplication of two basis elements $\beta_i$ and $\beta_j$ produces two terms – $\beta_{i+j}$ and $\beta_{|i-j|}$. The first term suggests an algorithm which in part is similar to the multiplication algorithm using polynomial bases where $\alpha^i\alpha^j = \alpha^{i+j}$.

The product of two elements $\mathbf{a}$ and $\mathbf{b}$ can be written as

$$\mathbf{c} = \mathbf{ab} = a_0\mathbf{b} + b_0\sum_{i=1}^{n-1} a_i\beta_i + \sum_{j=1}^{n-1} a_j(\sum_{i=1}^{n-1} b_i\beta_i\beta_j) \tag{7}$$

In the multiplication algorithm, the first term in (7) is performed by initialization. Assigning $B^+[0] = b_0$ at initialization computes $b_0\sum_{i=1}^{n-1} a_i\beta_i$ in the main loop. The last term is computed in the main loop as follows:

$$A_j = a_j\sum_{i=1}^{n-1} b_i\beta_i\beta_j = \underbrace{a_j\sum_{i=1}^{n-1} b_i\beta_{i+j}}_{B^+} + \underbrace{a_j\sum_{i=1}^{n-1} b_i\beta_{|i-j|}}_{B^-} \tag{8}$$

Algorithm 1 shows the multiplication over the binary extension field $\mathbb{F}_{2^n}$. When it is not specified, the registers are initialized with zero. The terms marked with $B^+$ and $B^-$ in (8) are computed in variables with the same names in the algorithm. The main loop is basically for computation and summing up of $A_j$'s for $j = 1 \ldots n-1$.

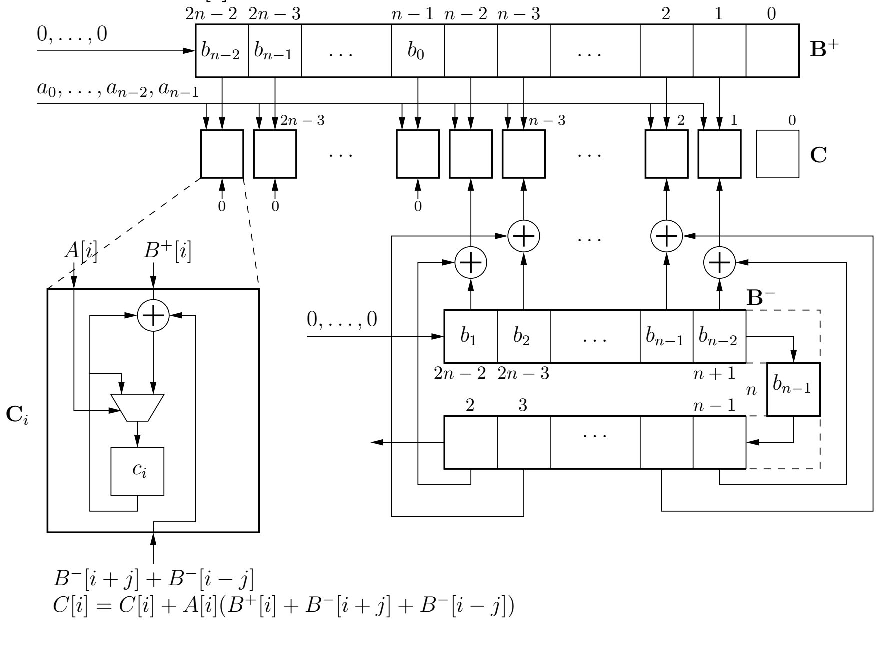
A hardware architecture for the sequential operation of the multiplier over $\mathbb{F}_{2^n}$ is shown in Fig. 1. The registers are initialized as shown in Algorithm 1 (The register $B^-$ is bent in the figure to show the operation clearly). Similar to bit-serial finite field multiplication architectures where a polynomial basis is used, the $a_i$ line drives $n$ gates. If the multiplier is implemented as shown in Fig. 1, this high capacitive load will decrease the maximum operating clock frequency of the system. Other than that, the fan-out of gates in Fig. 1 are very moderate. In the architecture there is no connection on $C[0]$ because this bit equals to $a_0b_0$ and is computed in the initialization phase.

**Algorithm 1** Two-pass sequential multiplication

```
Input: $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{2^n}$ where $\mathbf{a} = a_0 + \sum_{i=1}^{n-1} a_i\beta_i, \mathbf{b} = b_0 + \sum_{i=1}^{n-1} b_i\beta_i$
Output: $C = \mathbf{ab}$
Variables: $B^+[2n-2\dots0]$, $B^-[2n-2\dots2]$, $C[2n-2\dots0]$
  { initialization }
  $B^+[2n-2\dots n-1] \leftarrow (b_{n-1},\dots,b_1,b_0)$
  $B^-[2n-2\dots n] \leftarrow (b_1,b_2,\dots,b_{n-1})$
  if $a_0 = 1$ then
    $C[n-1\dots0] \leftarrow (b_{n-1},\dots,b_0)$
  end if

  { main loop }
  for $i = n-1$ to $1$ do
    if $a_i = 1$ then
      $C[2n-2\dots1] \leftarrow C[2n-2\dots1] + B^+[2n-2\dots1]$
      $C[n-2\dots1] \leftarrow C[n-2\dots1] + B^-[2\dots n-1] + B^-[2n-2\dots n+1]$
    end if
    shift right $B^+$
    shift right $B^-$
  end for
  return $C$
```

**Fig. 1.** Hardware architecture for the two-pass sequential multiplication algorithm and the $C[i]$ module



**Step 2– Reduction** Based on (4) we obtain $\beta_{2n-i} = \beta_{n-i}\beta_n - \beta_i$ where $2 \le i \le n-1$. Given the irreducible polynomial in (6),

$$\beta_{2n-i} = -(\sum_{j=1}^{n-1} r_j\beta_{n-i}\beta_j + r_0\beta_{n-i}) - \beta_i = t^+ + t^-$$

where

$$\begin{aligned} t^+ = r_{n-1}\beta_{(2n-i)-1} + r_{n-2}\beta_{(2n-i)-2} + \ldots + r_3\beta_{(2n-i)-(n-3)} \\ +r_3\beta_{(2n-i)-(n-2)} + r_1\beta_{(2n-i)-(n-1)} + r_0\beta_{n-i} \end{aligned} \tag{9}$$

and

$$t^- = r_{n-1}\beta_{|i-1|} + r_{n-2}\beta_{|i-2|} + \ldots + r_3\beta_{|n-i-3|} + r_2\beta_{|n-i-2|} + r_1\beta_{|n-i-1|} \tag{10}$$

The architecture in Fig. 1 can be used for reduction if $B^+$ and $B^-$ registers are initialized properly with the coefficients of the irreducible polynomial (6) and $c_{2n-2}, c_{2n-3}, \ldots, c_n$ are applied to the system instead of $a_{n-1}, \ldots, a_2, a_1$ . This can be verified by comparing Algorithms 2 to 1. The control algorithm for the hardware is different in multiplication and reduction steps.

Equation (9) implies that in order to reduce $\beta_{2n-1}$, one needs to set the coefficients of the irreducible polynomials in the location $2n-i-1$ to $n-i$ of the register $B^+$ and add it to the contents of $C$. Implementation of (10) is performed by by proper initialization of the $B^-$ register as shown in Algorithm 2. When $\mathbf{r} = \beta_n + 1$, multiplication and reduction steps can be efficiently merged together [1].

## 4 An Iterative Algorithm for Multiplication

Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{p^n}$ be represented using the Dickson basis as before. The product of these two elements $\mathbf{c} = \mathbf{ab}$ is written as

$$\mathbf{c} = \mathbf{ab} = \mathbf{a}b_0 + \sum_{i=1}^{n-1} b_i(\mathbf{a}\beta_i). \tag{11}$$

Based on (5) we can write,

$$\mathbf{a}\beta_{i+1} = \beta_1(\mathbf{a}\beta_i) - \mathbf{a}\beta_{i-1} \tag{12}$$

**Algorithm 2** Reduction

**Input:** $\mathbf{c} \in \mathbb{F}_{2^n}$ where $\mathbf{c} = c_0 + \sum_{i=1}^{2n-2} c_i\beta_i$ and the coefficients of the irreducible polynomial $\mathbf{r} = \beta_n + \sum_{i=1}^{n-1} r_i\beta_i + r_0$
**Output:** $\mathbf{c}\ (mod\, \mathbf{r})$
**Variables:** $B^+[2n-2\ldots 0]$, $B^-[n-2\ldots -(n-2)]$, $C[2n-2\ldots 0]$

```
{ initialization }
$B^+[2n-2\ldots n-2] \leftarrow (r_n - 1, \ldots, r_1, r_0)$ {assign $r_0$ as well, and this slightly simplifies the algorithm compared to [1]}
$B^-[n-2\ldots 0] \leftarrow (r_1, \ldots, r_{n-2}, r_{n-1})$
$B^-[-1] \leftarrow 1$ {set this bit}
$C[2n-2\ldots 0] \leftarrow (c_{2n-2}, \ldots, c_1, c_0)$
{ main loop }
for $i = 2n-2$ to $n$ do
  shift right $B^+$
  shift right $B^-$
  if $C[i] = 1$ then
    $C[2n-2\ldots 1] \leftarrow C[2n-2\ldots 1] + B^+[2n-2\ldots 1]$
    $C[n-2\ldots 1] \leftarrow C[n-2\ldots 1] + B^-[2\ldots n-1] + B^-[2n-2\ldots n+1]$
  end if
end for
return $C[n-1\ldots 0]$
```

which gives a relation to iteratively compute $\mathbf{a}\beta_i$ in (11). Equation (12) requires multiplication by $\beta_1$ which is implemented as follows. Let $\mathbf{t} \in \mathbb{F}_{p^n}$,

$$\mathbf{t}\beta_1 = (\sum_{i=1}^{n-1} t_i\beta_i\beta_1) + t_0\beta_1$$

applying (5),

$$\begin{aligned}\mathbf{t}\beta_1 = t_{n-1}\beta_n + t_{n-2}\beta_{n-1} + t_{n-3}\beta_{n-2} + \ldots + t_2\beta_3 + t_1\beta_2 + t_0\beta_1 \\ +t_{n-1}\beta_{n-2} + \ldots + t_4\beta_3 + t_3\beta_2 + t_2\beta_1 + t_1\beta_0 \\ = t_{n-1}\beta_n + t_{n-2}\beta_{n-1} + (\sum_{i=1}^{n-2}(t_{i-1} + t_{i+1})\beta_i) + t_1\beta_0\ . \quad (13)\end{aligned}$$

Given the irreducible polynomial $\mathbf{r} = \beta_n + \sum_{i=1}^{n-1} r_i\beta_i + r_0$, (13) is reduced to

$$\mathbf{t}\beta_1 = (t_{n-2} + t_{n-1}r_{n-1})\beta_{n-1} + (\sum_{i=1}^{n-2}(t_{i-1} + t_{i+1} + t_{n-1}r_i)\beta_i) + \quad (14)$$
$$t_{n-1}r_0 + t_1\beta_0.$$

Equation (14) can be implemented using combinational logic for an arbitrary irreducible polynomial $\mathbf{r}$ of the form (6). Algorithm 3 implements (11) using (12) and (14) over $\mathbb{F}_{2^n}$. Multiplication over $\mathbb{F}_{p^n}$ is similar. The algorithm uses $A_{prev}$, $A_{curr}$ and $A_{next}$ $n$-bit registers to hold the $n$ coefficients of $\mathbf{a}\beta_{i-1}, \mathbf{a}\beta_i$ and $\mathbf{a}\beta_{i-1}$ respectively, in each iteration (Over $\mathbb{F}_{p^n}$ each element of the registers holds an element of $\mathbb{F}_p$). The coefficients of the irreducible polynomial $\mathbf{r}$ are held in the $n$-bit register $R$. Coefficient $r_n$, which is assumed to be always one, is not saved. Function `MultBase`$(A, \beta_1)$, which implements (14), multiplies the input $A$ by $\beta_1$ and reduces the result *mod* the irreducible polynomial $\mathbf{r}$.

**Hardware Architecture** Figure 2(a) depicts a hardware architecture for the multiplier in which sequential logic and combinational logic (i.e., $n$-bit registers) are shown by rectangles and circles respectively. The $\underline{\times\beta_1}$ combinational logic, shown in Fig. 2(b) implements (14), where finite field addition and multiplication are performed in the ground field. The hardware needs to be initialized as stated at the beginning of Algorithm 3. It is possible to initialize $A_{curr}$ with $A\beta_1$ by one extra iteration at start up, but over $\mathbb{F}_{2^n}$, the architecture requires a little special consideration with one of its bits since $\beta_0 = 2 = 0 \, mod \, 2$. Similar to other serial implementation of finite field multipliers a high fan-out is required on the line which supplies $b_0, b_1, \ldots b_{n-1}$ to the system. A high fan-out is also required in the $\times\beta_1$ module for the $r_{n-1}$ line.

**Hardware Comparison** The hardware complexities, in terms of gate counts, of serial multipliers in Fig. 1 and 2 grow linearly with $n$, which is similar to the typical implementation of finite field multiplication using polynomial bases. The delay seems to be constant and independent to $n$. However, a fanout equal to $n$ is required which increases the critical path delay and/or the area. Table 2 compares the hardware resources required for sequential multiplication using Dickson bases and polynomial bases implemented over $\mathbb{F}_{2^n}$. The current values in Table 2 are for an arbitrary $\mathbf{r}$. When $\mathbf{r}$ is fixed, some of these would be reduced. For example, number of AND and XOR gates would be $n$ and $3n + H(\mathbf{r}) - 4$ respectively, where $H(\mathbf{r})$ is the Hamming weight of $\mathbf{r}$.

## 5 Conclusion

We have explained Dickson bases using simple mathematical terms and have used it to give a development of the computation model/algorithm of

**Algorithm 3** Iterative multiplication and reduction

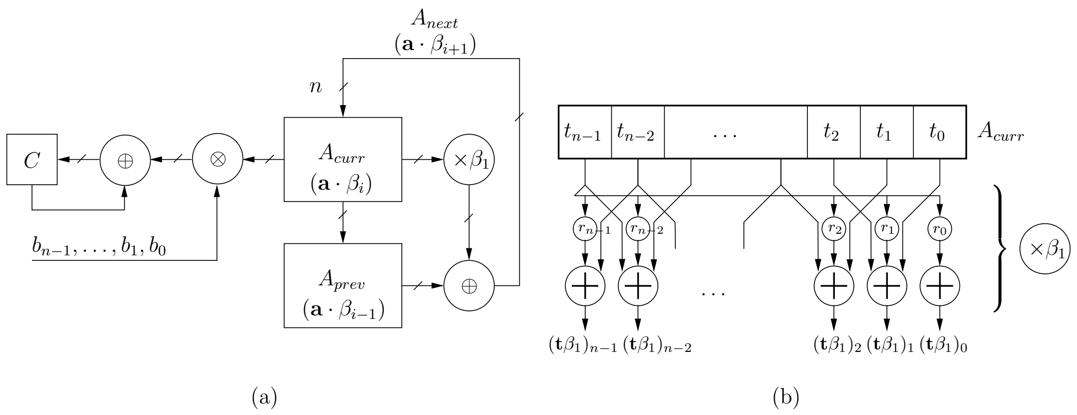
**Input:** $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{2^n}$ where $\mathbf{a} = a_0 + \sum_{i=1}^{n-1} a_i\beta_i$, $\mathbf{b} = b_0 + \sum_{i=1}^{n-1} b_i\beta_i$ and coefficients of the irreducible polynomial $\mathbf{r} = \beta_n + \sum_{i=1}^{n-1} r_i\beta_i + r_0$
**Output:** $C = AB \ (mod\, \mathbf{r})$

```
{ initialization }
$A \leftarrow (a_{n-1}, \dots, a_1, a_0)$
$R \leftarrow (r_{n-1}, \dots, r_1, r_0)$
$A_{prev} \leftarrow 0$
$A_{curr} \leftarrow \texttt{MultBase}(A, \beta_1)$
if $b_0 = 1$ then
  $C \leftarrow A$
else
  $C \leftarrow 0$
end if
{ main loop }
for $i = 1$ to $n - 1$ do
  $A_{next} \leftarrow A_{prev} + \texttt{MultBase}(A_{curr}, \beta_1)$
  if $b_i = 1$ then
    $C = C + A_{curr}$
  end if
  $A_{prev} \leftarrow A_{curr}$
  $A_{curr} \leftarrow A_{next}$
end for
return $C$

Function $\texttt{MultBase}(T, \beta_1)$
Input: register $T$ which contains $\mathbf{t} \in \mathbb{F}_{2^n}$, $\beta_1$ and the coefficients of the irreducible polynomial $R = (r_{n-1}, \dots, r_1, r_0)$
Output: $T = A\beta_1 \ (mod\, \mathbf{r})$
{When registeres are shifted 0's are inserted }
$V \leftarrow (T << 1) + (T >> 1)$
$V[0] \leftarrow 0$
{reduction}
if $T[n-1] = 1$ then
  $V \leftarrow V + R$
end if
return $V$
```

[1] and mapped it on a hardware architecture for field multiplication. We have also introduced an alternate multiplication algorithm and its hardware architecture, which are simpler than those based on the model presented in [1] and have compared the new multiplier to a typical sequential polynomial basis field multiplier. Our comparison results suggest that for multiplication using Dickson bases further innovation is needed to achieve the same level of hardware/arithmetic complexity as that of the corresponding multiplication using conventional polynomial bases.

**Fig. 2.** (a) Hardware architecture for iterative filed multiplication using the Dickson basis and (b) A circuit diagram for multiplying a field element by $\beta_1$



$A_{next}$
$(\mathbf{a}\cdot\beta_{i+1})$
$n$
$C$
$\oplus$
$\otimes$
$A_{curr}$
$(\mathbf{a}\cdot\beta_i)$
$\times\beta_1$
$b_{n-1},\ldots,b_1,b_0$
$A_{prev}$
$(\mathbf{a}\cdot\beta_{i-1})$
$\oplus$
$t_{n-1}$ $t_{n-2}$ $\ldots$ $t_2$ $t_1$ $t_0$
$A_{curr}$
$r_{n-1}$ $r_{n-2}$ $r_2$ $r_1$ $r_0$
$\times\beta_1$
$(\mathbf{t}\beta_1)_{n-1}$ $(\mathbf{t}\beta_1)_{n-2}$ $(\mathbf{t}\beta_1)_2$ $(\mathbf{t}\beta_1)_1$ $(\mathbf{t}\beta_1)_0$
(a)
(b)

**Table 2.** Comparison of hardware resources

| Multiplier | Flip Flop | Two input *AND* gate | Two input *XOR* gate | Delay | cycles | Max fan-out |
|---|---|---|---|---|---|---|
| Fig. 1 | $6n-6$ | $2n-1$ | $5n-3$ | $2T_{XOR}+T_{AND}$ | $2(n-1)$ [a] | $2(n-1)$ |
| Fig. 2 | $3n$ | $2n$ | $4n-3$ | $2T_{XOR}+T_{AND}$ | $n$ | $2\times n$ [b] |
| Using Polynomial Bases | $2n$ | $2n$ | $2n$ | $T_{XOR}+T_{AND}$ | $n$ | $n$ |

[a] When irreducible binomials are used or the irreducible polynomial is known, multiplication and reduction steps can be computed in $n-1$ cycles [1].

[b] Requires two high fan-out lines

## References

1. Ronald C. Mullin and Ayan Mahalanobis. Dickson bases and finite fields. Technical report, Center for Applied Cryptography Research, 2005.
2. Ronald C. Mullin, I. M. Onyszchuk, Scott A. Vanstone, and R. M. Wilson. Optimal normal bases in GF($p^n$). *Discrete Appl. Math.*, 22(2):149–161, 1989.
3. R. Lidl, Gary L. Mullen, and G. Turnwald. *Dickson Polynomials.* Longman Scientific and Technical, 1993.
4. Leonard Eugene Dickson. The analytic representation of substitution on a power of prime number of letters with a discussion of the linear group. *Ann. of Math.*, 1896/97.