Low Complexity Word-Level Sequential Normal Basis Multipliers

Arash Reyhani-Masoleh[1] and M. Anwar Hasan[2]

13th April 2004

Centre for Applied Cryptographic Research
[1] Department of Combinatorics and Optimization
[2] Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.
E-mails: areyhani@uwaterloo.ca and ahasan@ece.uwaterloo.ca

Abstract

For efficient hardware implementation of finite field arithmetic units, the use of a *normal basis* is advantageous. In this report, two classes of architectures for multipliers over the finite field $GF(2^m)$ are proposed. These multipliers are of sequential type, i.e., after receiving the coordinates of the two input field elements, they go through k, $1 \leq k \leq m$, iterations (i.e., clock cycles) to finally yield all the coordinates of the product in parallel. The value of k depends on the word size $w = \left\lceil \frac{m}{k} \right\rceil$. For $w > 1$, these multipliers are highly area efficient and require fewer number of logic gates even when compared with the most area efficient multipliers available in the open literature. This makes the proposed multipliers suitable for applications where the value of m is large but space is of concern, e.g., resource constrained cryptographic systems. Additionally, if the field dimension m is composite, i.e., $m = kn$, then the extension of one class of the architectures yields a highly efficient multiplier over composite fields.

Keywords: Finite field, Massey-Omura multiplier, optimal normal basis.

1 Introduction

Finite field $GF(2^m)$ is a set of 2^m elements where we can add, subtract, multiply and divide (by non-zero elements) without leaving the set. Arithmetic operations over finite fields are widely used in error control coding and cryptography. In both of these applications, there is a need to design low complexity finite field arithmetic units. The complexity of such a unit largely depends on how the field elements are represented, and there are many ways to represent field elements. Among them, representation of field elements using a normal basis is quite attractive for efficient hardware implementation. A normal basis exists for every extended finite field. Massey and Omura are the first to propose multipliers based on the normal basis [1].

Like any finite field multiplier, a hardware implementation of a normal basis multiplier can be categorized either as a parallel or sequential type. In a typical parallel multiplier for $GF(2^m)$, once $2m$ bits of two inputs are received, m bits of the product are obtained together at the output after delays through various logic gates (if the multiplier is implemented using combinational logic gates) or after delays due to a memory access (if the multiplier is implemented using a look-up table). Such a parallel type multiplier (see for example [2, 3, 4]) requires a lot of silicon area and is considered to be impractical for cryptographic applications where finite fields with very large values of m (e.g., 4000) are used.

On the other hand, a bit-level sequential multiplier is much (about m times) more area efficient, but in general takes m iterations (or clock cycles) for one multiplication. Some sequential multipliers generate one bit of the product in each of these m cycles. In another type of sequential multipliers, all m bits of the product are incrementally generated for $m-1$ cycles and become the final form of the product at the end of the m-th cycle. These two types multipliers are hereafter referred to as sequential multipliers with serial output (SMSO) and sequential multipliers with parallel output (SMPO), respectively. Examples of the former type includes Berlekamp's bit-serial dual basis multiplier [5] and Massey-Omura's original bit-serial normal basis multiplier [1], while those of the latter type includes the normal basis

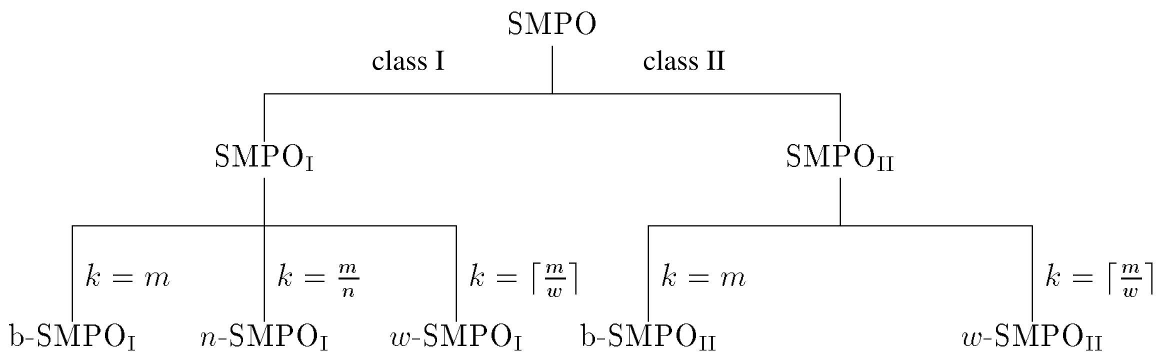

Figure 1: The representation of the proposed SMPO architectures.

multiplier due to Agnew et al. [6] and the well known polynomial basis multiplier based on LFSR [7]. Usually, SMPO multipliers run at a much higher clock rate than their SMSO counterparts.

In this report, we propose two new classes of word-level SMPO multiplier architectures using a normal basis. These two classes of structures are hereafter referred to as SMPO$_\mathrm{I}$ and SMPO$_\mathrm{II}$. They take k, $1 \le k \le m$, cycles to complete the multiplication and their critical path delay is proportional to $\lceil \log_2 \frac{m}{k} \rceil$. The maximum word size can be chosen based on the available chip area. As a starting point, for one-bit long words, i.e., $k = m$, we simply present bit-level SMPO structures which are referred to as b-SMPO$_\mathrm{I}$ and b-SMPO$_\mathrm{II}$. The AND gate count for the b-SMPO$_\mathrm{I}$ structure is $\lfloor \frac{m}{2} \rfloor + 1$ only. This implies that if the multiplication over $GF(2^m)$ is performed using a suitable subfield $GF(2^n)$ where $n > 1$ and $m = nk$, then the corresponding architecture (which is referred to as n-SMPO$_\mathrm{I}$) will yield a highly efficient multiplier for composite fields. To the best of our knowledge, no such AND efficient bit-level $GF(2^m)$ multiplier, i.e., b-SMPO$_\mathrm{I}$, in other field representations such as polynomial, dual or redundant basis exists. We then extend the SMPO structures to word size of w bits, where $2 \le w \le m$ and m is not a multiple of w, i.e., $k = \lceil \frac{m}{w} \rceil$, $1 \le k \le \lceil \frac{m}{2} \rceil$. These structures are denoted as w-SMPO$_\mathrm{I}$ and w-SMPO$_\mathrm{II}$. In this report, the gate counts of the proposed architectures are also given and it is shown that they are better than those of the existing architectures. Figure 1 depicts the classification of the proposed five multiplier architectures under two classes–SMPO$_\mathrm{I}$ and SMPO$_\mathrm{II}$.

The organization of the remainder of this report is as follows. In Section 2, some preliminaries regarding the normal basis are given and two well-known bit-level sequential multipliers are briefly reviewed. Then in Section 3, we propose our bit-level SMPO architectures, i.e., b-SMPO$_{\mathrm{I}}$ and b-SMPO$_{\mathrm{II}}$. In this section, we also compare their complexities with those of similar structures available in the open literature. Moreover, the complexities are compared for five specific fields recommended by the National Institute of Standards and Technology (NIST) for elliptic curve digital signature algorithm (ECDSA). In Section 4, our new b-SMPO$_{\mathrm{I}}$ structure is extended to n-SMPO$_{\mathrm{I}}$ for composite $m = kn$. Its complexities are compared with the best known architectures for arbitrary composite m. We also consider those composite values of m recommended by ANSI X.62 for ECDSA[1]. In Section 5, our new word-level SMPO architectures (w-SMPO$_{\mathrm{I}}$ and w-SMPO$_{\mathrm{II}}$) are given and their complexities are compared for general and optimal normal bases. Finally, conclusions are made in Section 6.

2 Preliminaries

2.1 Normal Basis Representation and Multiplication

Let a normal basis of $GF(2^m)$ over $GF(2)$ be

$$N = \{\beta, \beta^2, \cdots, \beta^{2^{m-1}}\},$$

where $\beta \in GF(2^m)$. It is well known that there exists such a normal basis for any positive integer $m > 1$ [8]. Using such a basis, any field element $A \in GF(2^m)$ can be represented as a linear combination of the elements of N, i.e., $A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = (a_0, a_1, \cdots, a_{m-1})$, where $a_i \in GF(2)$, $0 \le i \le m-1$, are referred to as coordinates of A with respect to N. In hardware implementation, A^2 is almost free of cost and can be easily performed by right cyclic shifts, i.e., $A^{2^i} = (a_{m-i}, a_{m-i+1}, \cdots, a_{m-i-1})$. However, multiplication is not as easy

[1]NIST recommended values of m for ECDSA are not composite.

as squaring.

Let $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$ be two elements of $GF(2^m)$, where a_i's and b_i's are their respective normal basis coordinates. Let $C = (c_0, c_1, \cdots, c_{m-1})$ be their product: $C = AB$. Then, any coordinates of C, say c_{m-1}, is a function u of A and B which can be obtained by a matrix multiplication, i.e., $c_{m-1} = u(A, B) = \mathbf{a} \cdot \mathbf{M} \cdot \mathbf{b}^T$, where $\mathbf{M}$ is a binary $m \times m$ matrix known as the *multiplication matrix* [9], $\mathbf{a} = [a_0, a_1, \cdots, a_{m-1}]$, $\mathbf{b} = [b_0, b_1, \cdots, b_{m-1}]$ and T denotes vector transposition. The number of 1's in $\mathbf{M}$ is known as the *complexity* of the normal basis N [10] and is denoted as C_N. The latter determines the gate counts and hence time delay for a normal basis multiplier.

Remark 1. *It is well known that $C_N \geq 2m - 1$. If $C_N = 2m - 1$, then the normal basis is called an optimal normal basis.*

Most of the existing word-level multipliers are SMSO type. In this report, we will present two new classes of SMPO architectures. Below we briefly review bit-serial multipliers due to Massey-Omura [1] and Agnew et al. [6]. The former, which is believed to be the first normal basis multiplier reported in the open literature, is a sequential multiplier with serial output (SMSO) whereas the latter is a sequential multiplier with parallel output (SMPO).

2.2 Bit-Level Sequential Multiplier with Serial Output

In [1], Massey and Omura have shown that if the function $u(A, B)$ is implemented to generate c_{m-1}, then the other coordinates of C can be obtained from the same implementation with inputs appropriately shifted in cyclic fashion, more precisely $c_{m-1-i} = u(A^{2^i}, B^{2^i})$. A block diagram of such an architecture for SMSO is presented in Figure 2(a). In this figure, all coordinates of A and B are first serially loaded into the shift registers. Then in each clock cycle, one coordinate of C from c_{m-1} to c_0 is generated by the u function just by cyclic shifts of the registers. The following example is used to illustrate the complexity of the u function. The field and the normal basis presented in this example will be used in all architectures presented in this report.

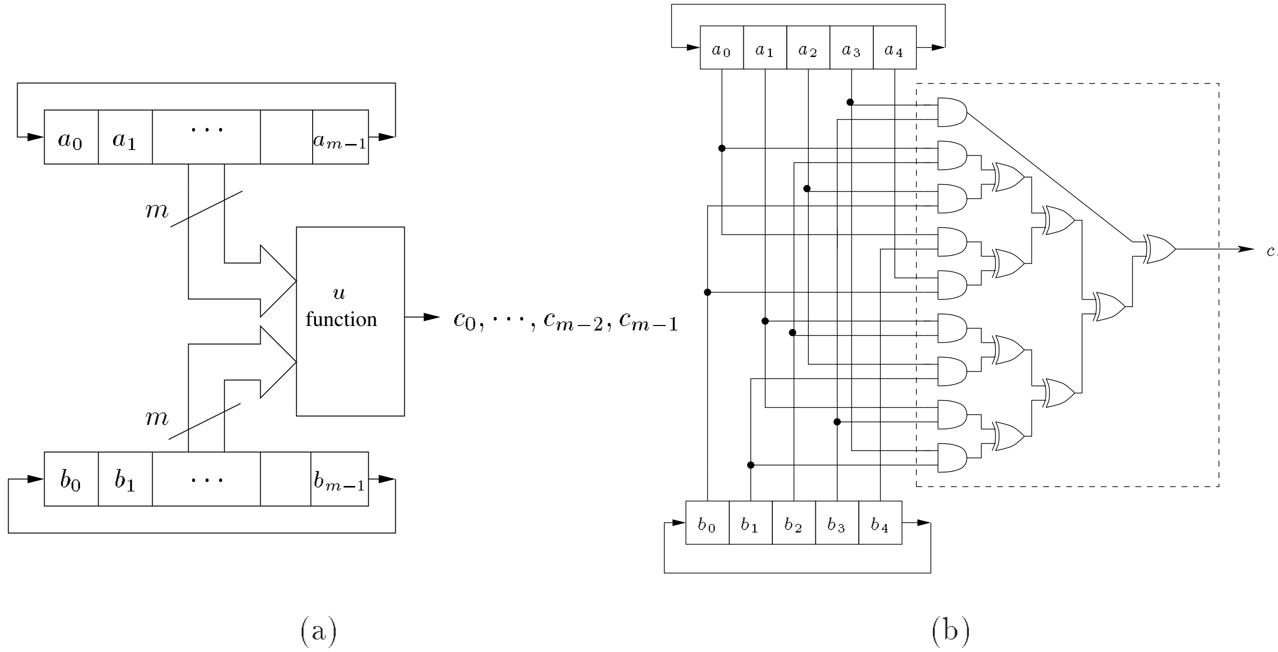

Figure 2: (a) Massey-Omura Bit-level SMSO for $GF(2^m)$. (b) $GF(2^5)$ Massey-Omura multiplier of Example 1.

Example 1. *Consider the finite field $GF(2^5)$ generated by the irreducible polynomial z^5+z^2+1 and let α be its root. If we choose $\beta = \alpha^5$, then it can be verified that $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}\}$ is a normal basis. Now, using Table 2 in [10], we have*

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$c_4 = a_3b_3 + (a_0b_2 + a_2b_0) + (a_0b_4 + a_4b_0) + (a_1b_2 + a_2b_1) + (a_1b_3 + a_3b_1), \tag{1}$$

and corresponding $GF(2^5)$ bit-serial multiplier is shown in Figure 2(b).

In general, the number of AND gates and XOR gates of Figure 2(a) are C_N and $C_N - 1$, respectively. Also, its critical path delay is $T_A + \lceil \log_2 C_N \rceil T_X$, where T_A and T_X are the time delays due to one AND gate and one XOR gate, respectively.

It is well known that (1) can be rearranged to reduce the AND gate count of the Massey-Omura multiplier from C_N to m (see for example [11]). This increases the critical path of the multiplier from $T_A + \lceil \log_2 C_N \rceil \, T_X$ to $T_A + (\lceil \log_2 \rho \rceil + \lceil \log_2 m \rceil) T_X$, where ρ is the maximum number of 1's among all rows (or columns) of the multiplication matrix $\mathbf{M}$. For an optimal normal basis, $\rho = 2$ and $C_N = 2m - 1$. Thus, the difference in the critical path delays for these two variants of the Massey-Omura multipliers disappears when an optimal normal basis is chosen. For tradeoff between area and time, one can use digit serial multiplier (see for example [12]).

2.3 Bit-Level Sequential Multiplier with Parallel Output

In [6], Agnew et al. presented another architecture for multiplier using the normal basis. The output coordinates of this multiplier are generated in parallel after m clock cycles (i.e., it is a bit-level SMPO architecture). For the field and normal basis constructed in Example 1, the corresponding multiplier architecture is shown in Figure 3(a). In this multiplier structure, all coordinates c_i, $0 \leq i \leq 4$ are obtained using (1) as follow

$$c_i = b_i a_{i+1} + b_{i+1}(a_i + a_{i+3}) + b_{i+2}(a_{i+3} + a_{i+4}) + b_{i+3}(a_{i+1} + a_{i+2}) + b_{i+4}(a_{i+2} + a_{i+4}), \quad (2)$$

where the additions in the subscript indices are reduced modulo 5. In (2), if one implements the first term, i.e., $b_0 a_1$ for c_0, the second term, i.e., $b_2(a_1 + a_4)$ for c_1 and up to the final term, i.e., $b_3(a_1 + a_3)$ for c_4, the SMPO of Figure 3(a) is obtained. The initial contents of shift registers A and B are shown in the figure. Details of the R_i cell are shown in Figure 3(b). Initially, the D_i latches of R_i's are cleared to zero and after m clock cycles the D_i's contain the coordinates of $C = AB$.

The number of AND gates and XOR gates of the SMPO can be easily obtained as m and C_N, respectively. The critical path delay of the multiplier is $T_A + (1 + \lceil \log_2 \rho \rceil) T_X$, where ρ is the maximum number of a_i terms that are XORed before being multiplied with a b_i term

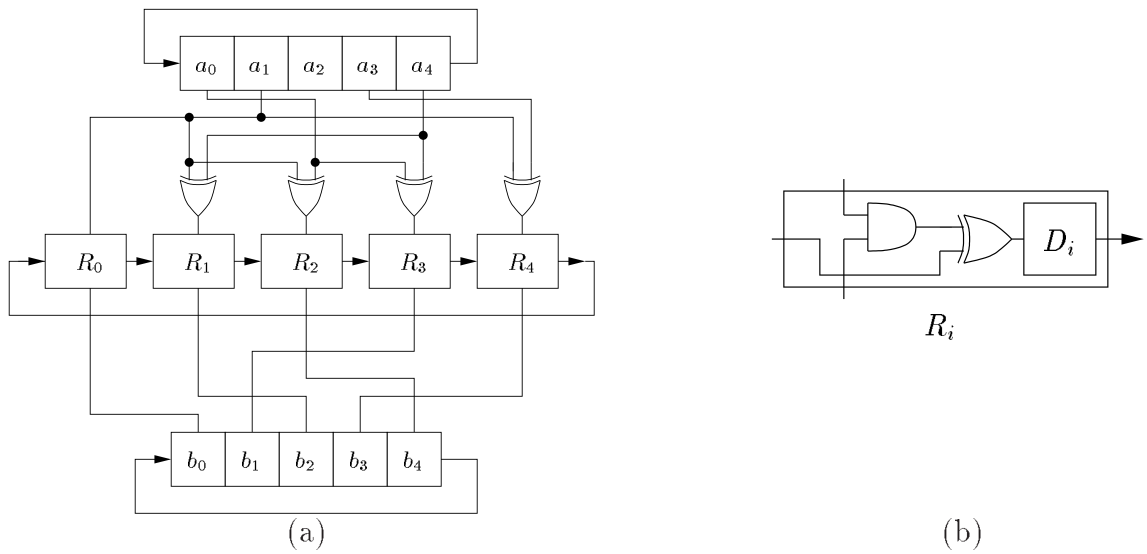

Figure 3: (a) $GF(2^5)$ SMPO due to Agnew et al. (b) Details of the R_i cell.

in (2). As mentioned earlier, this parameter ρ is the maximum number of 1's among all rows (or columns) of the multiplication matrix $\mathbf{M}$. It is noted that $2 \leq \rho \leq m$. For optimal normal bases (which is the case in Example 1), the critical path delay is $T_A + 2T_X$ as shown in Figure 3(a) and for an arbitrary normal basis, this delay is $\leq T_A + (1 + \lceil \log_2 m \rceil)T_X$.

2.4 Useful Lemmas

Before presenting our new architectures, below we present Lemmas 1 and 2 from [4] and [13], respectively. These lemmas will be used to formulate a multiplication algorithm which will then lead to different architectures.

Lemma 1. *[4] Let C be the multiplication of A and B over $GF(2^m)$, then*

$$C = \begin{cases} \sum_{i=0}^{m-1} a_i b_i \beta^{2^{i+1}} + \sum_{i=0}^{m-1} \sum_{j=1}^{v} x_{i,j} \delta_j^{2^i}, & \textit{for } m \textit{ odd} \\ \sum_{i=0}^{m-1} a_i b_i \beta^{2^{i+1}} + \sum_{i=0}^{m-1} \sum_{j=1}^{v-1} x_{i,j} \delta_j^{2^i} + \sum_{i=0}^{v-1} x_{i,v} \delta_v^{2^i}, & \textit{for } m \textit{ even,} \end{cases} \tag{3}$$

where $x_{i,j} = a_i b_{i+j} + a_{i+j} b_i$, $0 \leq i \leq m-1$, $1 \leq j \leq v$.

Lemma 2. *[13] Let C be the multiplication of A and B over $GF(2^m)$, then*

$$C = \begin{cases} \sum_{i=0}^{m-1} a_i b_i \beta^{2^i} + \sum_{i=0}^{m-1} \sum_{j=1}^{v} y_{i,j} \delta_j^{2^i}, & \textit{for } m \textit{ odd} \\ \sum_{i=0}^{m-1} a_i b_i \beta^{2^i} + \sum_{i=0}^{m-1} \sum_{j=1}^{v-1} y_{i,j} \delta_j^{2^i} + \sum_{i=0}^{v-1} y_{i,v} \delta_v^{2^i}, & \textit{for } m \textit{ even,} \end{cases} \tag{4}$$

where $y_{i,j} = (a_i + a_{i+j})(b_i + b_{i+j}),\ 0 \le i \le m-1,\ 1 \le j \le v.$

For proofs, the reader is referred to [4] and [13].

3 Proposed Bit-Level SMPO Architectures

3.1 Formulation

As before, let us consider two $GF(2^m)$ elements $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$ and let their product be $C = (c_0, c_1, \cdots, c_{m-1})$. For $0 \le i \le m-1$, let

$$F_i(A, B) = a_{i-g} b_{i-g} \beta + \sum_{j=1}^{v} z_{i,j} \delta_j \tag{5}$$

where $v = \left\lfloor \frac{m}{2} \right\rfloor$, $\delta_j = \beta^{1+2^j}$, $1 \le j \le v$, and $g \in \{0, 1\}$ determines $z_{i,j}$ as follows. For $1 \le j < \left\lceil \frac{m}{2} \right\rceil$,

$$z_{i,j} = \begin{cases} (a_i + a_{i+j})(b_i + b_{i+j}), & \text{if } g = 0, \\ a_i b_{i+j} + a_{i+j} b_i, & \text{if } g = 1. \end{cases} \tag{6}$$

For even values of m, we have

$$z_{i,v} = \begin{cases} b_i(a_i + a_{i+v}), & \text{if } g = 0, \\ a_i b_{i+v}, & \text{if } g = 1. \end{cases} \tag{7}$$

Note that additions and subtractions in the above subscripts are reduced modulo m.

Now, we can state the following theorem which is the key equation for our new architectures.

Theorem 1. *Consider three elements A, B and $C = AB$ of $GF(2^m)$. Then,*

$$C = (((F_{m-1}^2 + F_{m-2})^2 + F_{m-3})^2 + \cdots + F_1)^2 + F_0 \tag{8}$$

where $F_i \in GF(2^m)$ is a short form of $F_i(A, B)$ as defined in (5).

Proof. Combining (3) and (4), one can obtain

$$C = \sum_{i=0}^{m-1} a_{i-g} b_{i-g} \beta^{2^i} + \sum_{i=0}^{m-1} \sum_{j=0}^{v} z_{i,j} \delta_j^{2^i},$$

where $z_{i,j}$ was previously defined in terms of g. Thus,

$$\begin{aligned} C &= \sum_{i=0}^{m-1} a_{i-g} b_{i-g} \beta^{2^i} + \sum_{i=0}^{m-1} \sum_{j=0}^{v} z_{i,j} \delta_j^{2^i} \\ &= \sum_{i=0}^{m-1} \left(a_{i-g} b_{i-g} \beta + \sum_{j=0}^{v} z_{i,j} \delta_j \right)^{2^i} \\ &= \sum_{i=0}^{m-1} F_i^{2^i} \\ &= (((F_{m-1}^2 + F_{m-2})^2 + F_{m-3})^2 + \cdots + F_1)^2 + F_0. \end{aligned}$$

□

In order to efficiently implement a normal basis multiplier based on Theorem 1, the following is useful.

Lemma 3. *For $F_i(A, B)$ as defined in (5), one has*

$$F_i(A, B) = F_{k+i}(A^{2^k}, B^{2^k}), \quad 0 \leq k, i \leq m-1.$$

Proof. Let us denote $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$, then the coordinates of A^{2^k} and B^{2^k} can be obtained by k-fold right cyclic shifts, i.e., $A^{2^k} = (a_{-k}, a_{1-k}, \cdots, a_{m-k-1})$ and $B^{2^k} = (b_{-k}, b_{1-k}, \cdots, b_{m-k-1})$, where the indices reduced modulo m. Using (5), one can write $F_i(A, B) = a_{k+i-g-k} b_{k+i-g-k} \beta + \sum_{j=1}^{v} z_{k+i-k,j} \delta_j = F_{k+i}(A^{2^k}, B^{2^k})$ which completes the proof.

$\square$

Based on Theorem 1 and the above lemma, we can now have the following algorithm for normal basis multiplication.

Algorithm 1. (Normal Basis Multiplication)

Input: $A,\ B \in GF(2^m)$ given with respect to N

Output: $C = AB$

1. Initialize $X = (a_0,\ a_1, \cdots,\ a_{m-1})$, $Y = (b_0,\ b_1, \cdots,\ b_{m-1})$, $D = (d_0,\ d_1, \cdots,\ d_{m-1}) = 0$
2. For $l = 1$ to m {
3. $\quad D = D^2 + F_{m-1}(X, Y)$ using (5)
4. $\quad X = X^2$ and $Y = Y^2$
5. }
6. $C = D$

In step 3 of this algorithm, we use the fixed function F_{m-1} with varying inputs, i.e., at the l-th iteration this function is $F_{m-1}(A^{2^{l-1}}, B^{2^{l-1}})$. By substituting $i = m - l$ and $k = l - 1$ into Lemma 3, one can see that $F_{m-t} = F_{m-1}(A^{2^{l-1}}, B^{2^{l-1}})$. Thus, instead of using F_{m-1}, $F_{m-2}, \cdots, F_0$ with fixed inputs A and B as shown in Theorem 1, the use of only F_{m-1} with varying inputs greatly simplifies implementation of Algorithm 1. This is discussed in the next section.

Although our subsequent discussions are centered around hardware architecture, we would like to make the following comments with regard to a possible software implementation of Algorithm 1. In software, one can generate $z_{i,j}$ on the fly to obtain $F_i(A, B)$ using pre-stored δ_j's which are fixed for a specific normal basis. In this situation, the algorithm corresponding to the b-SMPO$_{\mathrm{I}}$ will require fewer (compared to the b-SMPO$_{\mathrm{II}}$) number of instructions to be executed by the processor on which the algorithm is implemented. This is because in the truth table of $z_{i,j}$ in terms of a_i, a_{i+j}, b_i and b_{i+j}, there are four and six 1s for

$g = 0$, and $g = 1$, respectively. Hence $Pr\{z_{i,j} = 1|g = 0\} = \frac{1}{4}$ and $Pr\{z_{i,j} = 1|g = 1\} = \frac{3}{8}$ which results in a fewer number of XOR instructions on average.

3.2 Architectures

In order to map the above algorithm on hardware, the structure of Figure 4(a) is proposed. In the initialization step for the multiplication operation, the coordinates of A and B are serially loaded into the corresponding shift registers. This is similar to step 1 of Algorithm 1. Let $D(l) \in GF(2^m)$ denote the field element after the l-th iteration of step 3 whose normal basis coordinates are stored in m latches (denoted as $D_0\ D_1\ \cdots\ D_{m-1}$). Then, to start the multiplication operation, all D_i latches have to be cleared corresponding to step 1 of the algorithm. In Figure 4(a), the cyclic shift operation at the output of D_i's performs squaring to obtain D^2. Finally, the $D^2 + F_{m-1}$ of step 3 is realized by the Z array, the XOR array and an additional AND gate. The F block which is shown with dashed lines realizes F_{m-1} and will be used in Section 5 of this report. The Z array contains v number of Z blocks which generate $z_{m-1,j}$, $1 \leq j \leq v$, needed for F_{m-1} in (5) and the XOR array consists of XOR gates. Depending on the value of $g \in \{0, 1\}$, one of the two Z blocks as shown in Figures 4(b) and 4(c) is used. It is noted that for m even, the Z block for generating $z_{m-1,v}$ is different from both Figures 4(b) and 4(c). For this case, a slightly different Z block is needed which will generate $z_{m-1,v}$ corresponding to (7) and requires one AND gate and $g \in \{0, 1\}$ XOR gate. The multiplier architecture containing Z blocks as shown in Figure 4(b) and 4(c) are referred to as b-SMPO$_{\mathrm{I}}$ and b-SMPO$_{\mathrm{II}}$, respectively.

Example 2. *Here, we want to obtain the architectures of the b-SMPO$_{I/II}$ for the field and the basis constructed in Example 1. For this example,* $\delta_1 = \beta + \beta^{2^3}$ *and* $\delta_2 = \beta^{2^3} + \beta^{2^4}$ *and substituting these into (5) for* $i = m-1 = 4$*, we have* $F_4 = a_{4-g}b_{4-g}\beta + z_{4,1}(\beta + \beta^{2^3}) + z_{4,2}(\beta^{2^3} + \beta^{2^4})$*. Since the contents of the XOR array for both* $g = 0$ *and* $g = 1$ *are the same, here we only consider* $g = 0$*. Thus,* $F_4 = (a_4b_4 + z_{4,1}, 0, 0, z_{4,1} + z_{4,2}, z_{4,2})$*. Let* $D^2 = (d_4, d_0, d_1, d_2, d_3)$*, then the output of the XOR array would be* $D^2 + F_4 = (d_4 + a_4b_4 + z_{4,1}, d_0, d_1, d_2 + z_{4,1} + z_{4,2}, d_3 + z_{4,2})$

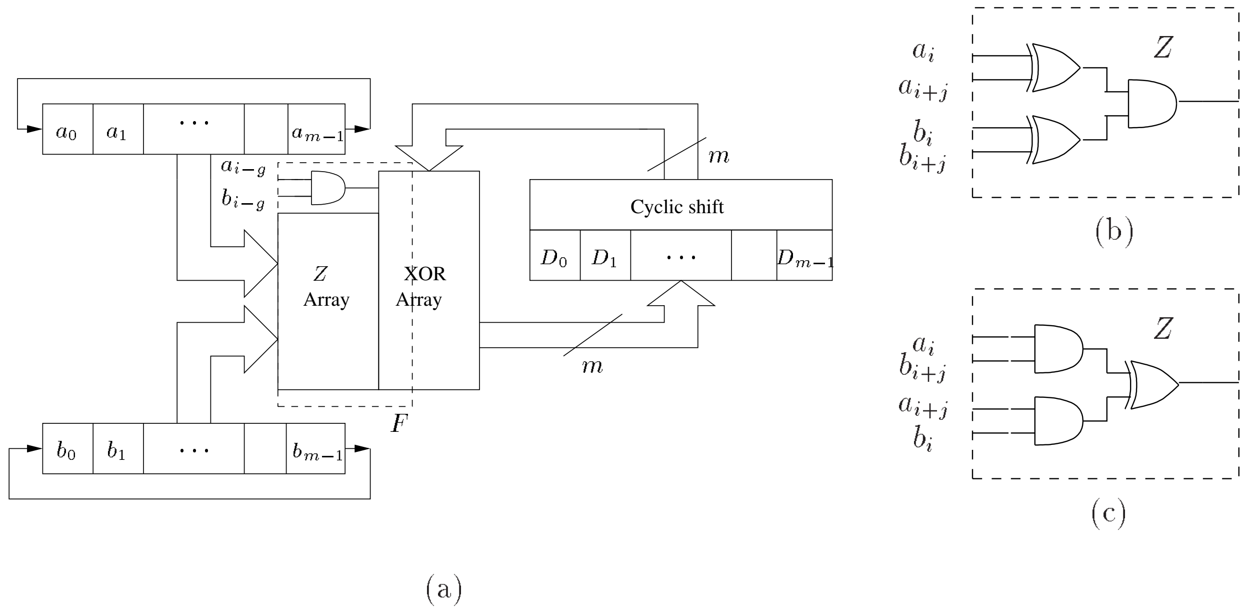

Figure 4: (a) The structure of the proposed b-SMPO$_{\mathrm{I/II}}$ over $GF(2^m)$. (b) The Z block for the b-SMPO$_{\mathrm{I}}$ ($g = 0$). (c) The Z block for the b-SMPO$_{\mathrm{II}}$ ($g = 1$).

which can be realized using 5 XOR gates. The architecture of b-SMPO$_I$ is shown in Figure 5. As seen in the figure, the F block which is shown with dashed lines generates F_4. The architecture of b-SMPO$_{II}$ for this example is similar to Figure 5 except that two Z blocks in the Z array and a_4b_4 in the first coordinate of F_4 should be replaced by Figure 4(b) and a_3b_3, respectively.

3.3 Complexities

The gate complexity of the b-SMPO$_{\mathrm{I/II}}$ depends on the number of gates in the Z array and the XOR array. The number of XOR gates and AND gates in the Z array are easy to obtain because it consists of v identical blocks[2]. These values for the proposed multipliers are shown in Table 1.

In general, if no terms or signals are reused, then the number of XOR gates in the XOR array of Figure 4 is upper bounded by $1+\sum_{j=1}^{v} H(\delta_j)$, where $H(\delta_j)$ is the number of non-zero coordinates in the normal basis representation of δ_j. For m being even, this value can be

[2]For m even, one block which generates $z_{m-1,v}$ is different from the others.

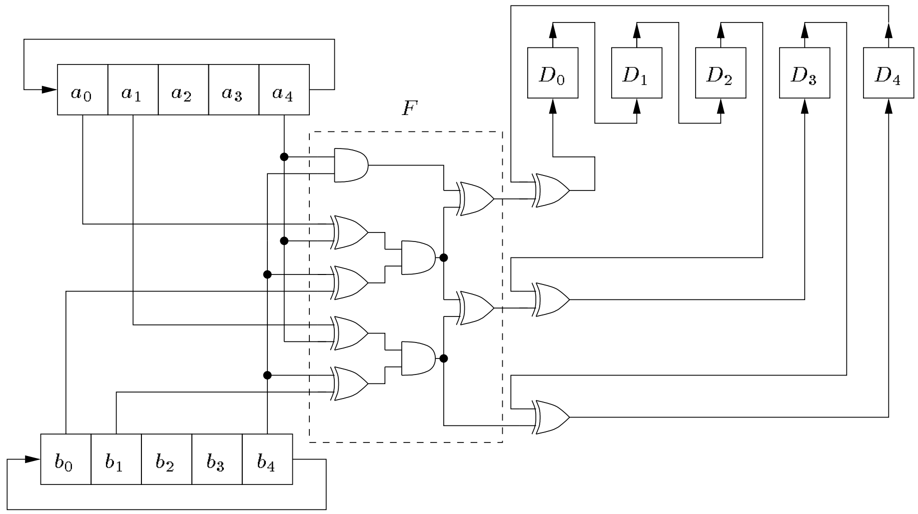

Figure 5: The proposed b-SMPO$_{\mathrm{I}}$ of Example 2.

Multiplier	#XOR	#AND
b-SMPO$_{\mathrm{I}}$	$m-1$	v
b-SMPO$_{\mathrm{II}}$	v	$m-1$

Table 1: The number of gates in the Z array.

reduced to $1+\sum_{j=1}^{v-1} H(\delta_j)+0.5H(\delta_v)$ by reusing half of the signals involved in δ_v [4]. Also, from [4]

$$C_N = \begin{cases} 1+2\sum_{j=1}^{v} H(\delta_j), & \text{for } m \text{ odd,} \\ 1+2\sum_{j=1}^{v-1} H(\delta_j)+H(\delta_v), & \text{for } m \text{ even.} \end{cases} \tag{9}$$

One can then conclude that the number of XOR gates in the XOR array is upper bounded by $0.5(C_N+1)$. Therefore, based on the above discussions, one can obtain the gate counts of the proposed multipliers as stated below.

Proposition 1. *The gate complexities of the proposed b-SMPO$_I$ and b-SMPO$_{II}$ are*

$$\begin{aligned} \#AND &= \left\lfloor \frac{m}{2} \right\rfloor + 1, \\ \#XOR &\leq \frac{C_N+2m-1}{2}, \end{aligned}$$

and

$$
\begin{aligned}
\#AND &= m, \\
\#XOR &\leq \frac{C_N + 1}{2} + \left\lfloor \frac{m}{2} \right\rfloor,
\end{aligned}
$$

respectively.

To obtain the maximum clock rate for the proposed multipliers, we obtain the delay of the critical path of the Z array and the XOR array in Figure 4(a). The delay of the Z array is $T_A + T_X$ for both $\mathrm{SMPO_I}$ and $\mathrm{SMPO_{II}}$. Since the output of the XOR array is

$$
D^2 + a_{i-g}b_{i-g}\beta + \sum_{j=1}^{v} z_{i,j}\delta_j, \tag{10}
$$

the critical path of the XOR array depends on the normal basis representations of δ_j, for $1 \leq j \leq v$. In the worst case when all δ_j's have a common coordinate, say β^{2^k}, then the critical path is determined by the k-th coordinate of the output of the XOR array and is equal to $\lceil \log_2(v+1) \rceil T_X$ for $k \neq 0$ or $\lceil \log_2(v+2) \rceil T_X$ for $k = 0$. Since $v = \left\lfloor \frac{m}{2} \right\rfloor$, one can easily verify that the critical path delay is upper bounded by $T_A + \lceil \log_2(m+4) \rceil T_X$. Let τ be the maximum number of terms among all m coordinates in normal basis representation of (10), then the critical path of the proposed multipliers would be $T_A + (1 + \lceil \log_2 \tau \rceil)T_X$. For optimal normal bases, it can be shown that $\tau = 3$ which makes the critical path of the XOR array to be $\lceil \log_2 3 \rceil T_X = 2T_X$ as shown in Figure 5.

Tables 2 and 3 compare our proposed bit-level sequential multipliers with the existing leading ones in terms of number of gates and latches as well as the critical path delay for generic and optimal normal basis.

For the purpose of illustration, in Table 4, we show the space and time complexities of the available bit-level sequential normal basis multipliers for the five binary fields recommended by NIST for ECDSA [17], where all fields are represented by type t Gaussian normal basis [18]. In this table, the total space column represents the total number of gates and latches.

Sequential Multiplier	#AND gates	#XOR gates	Total gate count generic N	Total gate count optimal N	#Latches	Output format
Massey-Omura[1]	C_N	C_N-1	$2C_N-1$	$4m-3$	$2m$	serial
IMO [11]	m	C_N-1	$m+C_N-1$	$3m-2$	$2m$	serial
Beth-Gollmann [14]	m	C_N	$m+C_N$	$3m-1$	$2m$	parallel
GG [15]	m	$\leq \frac{C_N+1}{2}+\left\lfloor\frac{m}{2}\right\rfloor$	$\leq \frac{C_N+2m+1}{2}+\left\lfloor\frac{m}{2}\right\rfloor$	$\leq 2m+\left\lfloor\frac{m}{2}\right\rfloor$	$3m$	parallel
Feng [16]	$2m-1$	$\leq C_N+m-1$	$\leq C_N+3m-2$	$\leq 5m-3$	$3m-2$	parallel
Agnew et al. [6]	m	C_N	$m+C_N$	$3m-1$	$3m$	parallel
b-SMPO$_{\mathrm{I}}$	$\left\lfloor\frac{m}{2}\right\rfloor+1$	$\leq \frac{C_N+2m-1}{2}$	$\leq \frac{C_N+2m+1}{2}+\left\lfloor\frac{m}{2}\right\rfloor$	$\leq 2m+\left\lfloor\frac{m}{2}\right\rfloor$	$3m$	parallel
b-SMPO$_{\mathrm{II}}$	m	$\leq \frac{C_N+1}{2}+\left\lfloor\frac{m}{2}\right\rfloor$	$\leq \frac{C_N+2m+1}{2}+\left\lfloor\frac{m}{2}\right\rfloor$	$\leq 2m+\left\lfloor\frac{m}{2}\right\rfloor$	$3m$	parallel

Table 2: Comparison of bit-level sequential normal basis multipliers over $GF(2^m)$. Note that for an optimal normal basis $C_N = 2m-1$, otherwise $C_N > 2m-1$. IMO=Improved Massey-Omura, GG=Geiselmann-Gollmann.

multiplier	generic N	optimal N	upper bound
Massey-Omura[1]	$T_A+\lceil\log_2 C_N\rceil T_X$	$T_A+(1+\lceil\log_2 m\rceil)T_X$	$\leq T_A+2\lceil\log_2 m\rceil T_X$
IMO [11]	$T_A+(\lceil\log_2\rho\rceil+\lceil\log_2 m\rceil)T_X$	$T_A+(1+\lceil\log_2 m\rceil)T_X$	$\leq T_A+2\lceil\log_2 m\rceil T_X$
Beth-Gollmann [14]	$T_A+(1+\lceil\log_2\rho\rceil)T_X$	T_A+2T_X	$\leq T_A+(1+\lceil\log_2 m\rceil)T_X$
GG [15]	$T_A+(1+\lceil\log_2\tau\rceil)T_X$	T_A+3T_X	$\leq T_A+\lceil\log_2(m+4)\rceil T_X$
Feng [16]	$T_A+(3+\lceil\log_2\rho\rceil)T_X$	T_A+4T_X	$\leq T_A+(3+\lceil\log_2 m\rceil)T_X$
Agnew et al. [6]	$T_A+(1+\lceil\log_2\rho\rceil)T_X$	T_A+2T_X	$\leq T_A+(1+\lceil\log_2 m\rceil)T_X$
b-SMPO$_{\mathrm{I}}$	$T_A+(1+\lceil\log_2\tau\rceil)T_X$	T_A+3T_X	$\leq T_A+\lceil\log_2(m+4)\rceil T_X$
b-SMPO$_{\mathrm{II}}$	$T_A+(1+\lceil\log_2\tau\rceil)T_X$	T_A+3T_X	$\leq T_A+\lceil\log_2(m+4)\rceil T_X$

Table 3: Comparison of critical path delays of bit-level sequential normal basis multipliers over $GF(2^m)$. Note that for an optimal normal basis $C_N = 2m-1$, otherwise $C_N > 2m-1$. Also, ρ and τ are defined in Subsections 2.3 and 3.3, respectively.

Using a C program, we have obtained the parameters ρ, τ, and C_N for these fields. Our findings show that $\rho = t$ and $\tau = t + 1$ for the recommended fields whose types (i.e., values of t) are given in the table.

In order to obtain a fast sequential multiplier, we need to reduce the number of clock cycles needed for the multiplication operation. In the following two sections, we consider this issue.

4 Extension to Composite Degree

Composite binary extension fields have received considerable attention in the recent past (see for example [19], [20], [21] and [22]). For such fields, the value of m is composite and special care should be taken in choosing such values for the cryptographic applications. In particular, in order to avoid the recent Weil descent attack on elliptic curve cryptosystems [23], [24], the reader is referred to references [25] and [26] for a secure family of composite fields.

When m is a multiple of k, i.e., $m = nk$ for an integer n, the proposed b-SMPO$_{\mathrm{I}}$ can be extended to an efficient multiplier[3] for the composite field $GF(2^{nk})$ by performing underlying operations over the subfield $GF(2^n)$. This subfield-level arithmetic based architecture is referred to as n-SMPO$_{\mathrm{I}}$. For such an extension, one needs to simply replace the AND and XOR gates shown in Figures 4(a) and 4(c) with the subfield multipliers and adders, respectively. Also, the three bit-level m-stage registers shown in Figure 4(a) are to be replaced by n-stage registers. Thus, in each clock cycle of the n-SMPO$_{\mathrm{I}}$ structure, one bit cyclic shift is replaced with one n-bit cyclic shift. As a result, the number of clock cycles required for the multiplication operation is reduced from m to $k = \frac{m}{n}$.

Table 5 compares the complexities of the proposed n-SMPO$_{\mathrm{I}}$ with the best ones available in the open literature, i.e., the multiple-bit structure proposed by Mullin [22] and the hybrid

[3] Note that the b-SMPO$_{\mathrm{II}}$ architecture is not suitable for such an extension as it requires more multiplications than that of b-SMPO$_{\mathrm{I}}$ architecture over the ground field, and such ground multiplications are usually costlier than additions.

m, t	Sequential Multiplier	#AND gates	#XOR gates	#Latches	Total space	Time delay
163, 4	Massey-Omura[1]	645	644	326	1615	$T_A + 10T_X$
	IMO [11]	163	644	326	1133	$T_A + 10T_X$
	Beth-Gollmann [14]	163	645	326	1134	$T_A + 3T_X$
	GG [15]	163	404	489	1056	$T_A + 4T_X$
	Feng [16]	325	807	487	1619	$T_A + 5T_X$
	Agnew et al. [6]	163	645	489	1297	$T_A + 3T_X$
	b-$\mathrm{SMPO_I}$	82	485	489	1056	$T_A + 4T_X$
	b-$\mathrm{SMPO_{II}}$	163	404	489	1056	$T_A + 4T_X$
233, 2	Massey-Omura[1]	465	464	466	1395	$T_A + 9T_X$
	IMO [11]	233	464	466	1163	$T_A + 9T_X$
	Beth-Gollmann [14]	233	465	466	1164	$T_A + 2T_X$
	GG [15]	233	349	699	1281	$T_A + 3T_X$
	Feng [16]	465	697	697	1859	$T_A + 4T_X$
	Agnew et al. [6]	233	465	699	1397	$T_A + 2T_X$
	b-$\mathrm{SMPO_I}$	117	465	699	1281	$T_A + 3T_X$
	b-$\mathrm{SMPO_{II}}$	233	349	699	1281	$T_A + 3T_X$
283, 6	Massey-Omura[1]	1677	1676	566	3919	$T_A + 11T_X$
	IMO [11]	283	1676	566	2525	$T_A + 12T_X$
	Beth-Gollmann [14]	283	1677	566	2526	$T_A + 4T_X$
	GG [15]	283	980	849	2112	$T_A + 4T_X$
	Feng [16]	565	1959	847	3371	$T_A + 6T_X$
	Agnew et al. [6]	283	1677	849	2809	$T_A + 4T_X$
	b-$\mathrm{SMPO_I}$	142	1121	849	2112	$T_A + 4T_X$
	b-$\mathrm{SMPO_{II}}$	283	980	849	2112	$T_A + 4T_X$
409, 4	Massey-Omura[1]	1629	1628	818	4075	$T_A + 11T_X$
	IMO [11]	409	1628	818	2855	$T_A + 11T_X$
	Beth-Gollmann [14]	409	1629	818	2856	$T_A + 3T_X$
	GG [15]	409	1019	1227	2655	$T_A + 4T_X$
	Feng [16]	817	2037	1225	4079	$T_A + 5T_X$
	Agnew et al. [6]	409	1629	1227	3265	$T_A + 3T_X$
	b-$\mathrm{SMPO_I}$	205	1223	1227	2655	$T_A + 4T_X$
	b-$\mathrm{SMPO_{II}}$	409	1019	1227	2655	$T_A + 4T_X$
571, 10	Massey-Omura[1]	5637	5636	1142	12415	$T_A + 13T_X$
	IMO [11]	571	5636	1142	7349	$T_A + 14T_X$
	Beth-Gollmann [14]	571	5637	1142	7350	$T_A + 5T_X$
	GG [15]	571	3104	1713	5388	$T_A + 5T_X$
	Feng [16]	1141	6207	1711	9059	$T_A + 7T_X$
	Agnew et al. [6]	571	5637	1713	7921	$T_A + 5T_X$
	b-$\mathrm{SMPO_I}$	286	3389	1713	5388	$T_A + 5T_X$
	b-$\mathrm{SMPO_{II}}$	571	3104	1713	5388	$T_A + 5T_X$

Table 4: Comparison among bit-level sequential normal basis multipliers over $GF(2^m)$ for the binary fields recommended by NIST for ECDSA.

Multiplier	# $GF(2^n)$ multipliers	# $GF(2^n)$ adders	#Latches	# Clock cycles	Type of basis
Mullin [22]	k	C_k	$3m$	k	normal
Paar et al. [21]	k	k	$2m+k$	k	polynomial
n-SMPO$_\mathrm{I}$	$\lfloor \frac{k}{2} \rfloor + 1$	$\leq \frac{C_k+2k-1}{2}$	$3m$	k	normal

Table 5: Comparison among the composite field multipliers of $GF(2^m)$ over $GF(2^n)$ where $m = nk$.

multiplier proposed by Paar et al. [21]. The data shown in the table for the architecture of [21] is for their optimized multiplier with $\gcd(n, k) = 1$. In this table, C_k denotes the complexity of the normal basis $GF(2^m)$ over $GF(2^n)$ and $C_k \geq 2k - 1$. As shown in this table, the n-SMPO$_\mathrm{I}$ architecture needs only $\lfloor \frac{k}{2} \rfloor + 1$ multiplications over the sub-field $GF(2^n)$ as compared to the others which require k multiplications. In practice, a parallel sub-field adder requires only n XOR gates, whereas a parallel sub-field multiplier requires n^2 AND gates and at most $n^2 + n$ XOR gates if either trinomial or pentanomial is used for sub-field multiplications [27]. However, if n is a composite number, one can use a multiplier structure proposed in [28] to obtain a fewer number of gates needed for the $GF(2^n)$ multiplier.

A number of composite fields are part of ANSI X9.62 [29]. For the purpose of illustrations, we have obtained the space complexity of n-SMPO$_\mathrm{I}$ for the composite fields recommended in the ANSI X9.62 standard for ECDSA. In this standard, $m \in \{176, 208, 272, 304, 368\}$, which can be written as $m = n \times k$, where $n = 16$ and $k \in K$,

$$K = \{11, 13, 17, 19, 23\}.$$

Since $\gcd(n, k) = 1$, the complexity of the normal basis $GF(2^m)$ over $GF(2^n)$ is the same as the one in $GF(2^k)$ over $GF(2)$ [8]. Thus, one can use Table 2 in [30] to obtain the lowest complexity of the normal basis of $GF(2^k)$ over $GF(2)$. These complexity values are $\{21, 45, 81, 117, 45\}$ for $k \in K$. It has been shown in [25] that these fields are secure against known attacks. Also, we use the polynomial basis multiplier proposed in [28] for sub-field operations over $GF(2^{16})$. Using Table 1 in [28], each $GF(2^{16})$ multiplier requires 144 AND

m (k, C_k)	word-serial multiplier	# AND gates	# XOR gates	#Latches	Total space complexity
	Mullin [22]	1584	3174	528	5286
176	Paar et al. [21]	1584	3014	363	4961
(11,21)	n-SMPO$_{\text{I}}$	864	1848	528	3240
	Mullin [22]	1872	4074	624	6570
208	Paar et al. [21]	1872	3562	429	5863
(13,45)	n-SMPO$_{\text{I}}$	1008	2324	624	3956
	Mullin [22]	2448	5682	816	8946
272	Paar et al. [21]	2448	4658	561	7667
(17,81)	n-SMPO$_{\text{I}}$	1296	3180	816	5292
	Mullin [22]	2736	6774	912	10422
304	Paar et al. [21]	2736	5206	627	8569
(19,117)	n-SMPO$_{\text{I}}$	1440	3752	912	6104
	Mullin [22]	3312	6654	1104	11070
368	Paar et al. [21]	3312	6302	759	10373
(23,45)	n-SMPO$_{\text{I}}$	1728	3744	1104	6576

Table 6: The space complexities of composite field multipliers for the fields recommended by ANSI X.62.

gates and 258 XOR gates which results in the number of gates shown in Table 6. In this table, total space complexity represents the addition of AND gates, XOR gates and latches. As seen in this table, the proposed n-SMPO$_{\text{I}}$ architecture has only 65-71 percent of the total space complexity of [21] and about 60% of that of [22].

5 New Word-Level SMPO Structures for Arbitrary Field Dimension

It follows from the discussions of the previous section that when $m = nk$, we can reduce the number clock cycles for the multiplication operation from m to k by performing underlying arithmetic over the subfield $GF(2^n)$. Each element of $GF(2^n)$ can be considered as an n-bit word. An interesting question is then "Is it possible to have such a fast multiplier using w-bit words where $w \nmid m$?". This is answered in the this section.

Recall that the multiplier structure given in Figure 4(a) realizes equation (8) in m clock cycles. For a w-bit word, $k = \left\lceil \frac{m}{w} \right\rceil$. To reduce this number from m to k, we need to realize w out of m terms in (8) in each clock cycle. Then, in each clock cycle, the three registers of Figure 4(a) should be shifted by w bits. Let $L_j(A, B) \in GF(2^m)$ be defined as

$$L_j(A, B) = \begin{cases} F_{m-1-jw}^{2^{w-1}} + F_{m-2-jw}^{2^{w-2}} + \cdots + F_{m-w-jw}, & \text{for } 0 \le j < k-1 \\ F_{w-1-r}^{2^{w-1}} + \cdots + F_0^{2^r}, & \text{for } j = k-1, \end{cases} \tag{11}$$

where r, $0 \le r \le w-1$, is obtained from $m = wk - r$ and F_i is as defined in (5). Then, using Theorem 1, the following can be verified by noting that L_j is the short form of $L_j(A, B)$.

Corollary 1. *Consider three elements A, B and $C = AB$ of $GF(2^m)$, where $m = wk - r$. Then,*

$$C^{2^r} = ((L_0^{2^w} + L_1)^{2^w} + \cdots + L_{k-2})^{2^w} + L_{k-1}, \tag{12}$$

where L_j is defined in (11).

Thus, instead of realizing F_{m-1} as we did in Figure 4(a), now we need to realize $L_0(A, B) = F_{m-1}^{2^{w-1}}(A, B) + F_{m-2}^{2^{w-2}}(A, B) + \cdots + F_{m-w}(A, B)$. Applying Lemma 3, we can realize L_0 just by using one function F_{m-1} with different cyclic shifts of inputs as follows:

$$L_0(A, B) = F_{m-1}^{2^{w-1}}(A, B) + F_{m-1}^{2^{w-2}}(A^2, B^2) + \cdots + F_{m-1}(A^{2^{w-1}}, B^{2^{w-1}}). \tag{13}$$

Based on the above discussion, an architecture for the word-level multiplier is shown in Figure 6. Like b-SMPO$_{\mathrm{I/II}}$, initially registers X and Y are loaded with the coordinates of $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$, respectively and the register D is cleared, i.e., $D = (0, 0, \cdots, 0)$. In this figure, each F block realizes the F_{m-1} function according to (5) and the total number of F blocks is w. These F blocks are denoted as $\mathcal{F}_i$, $0 \le i \le w-1$. Block CS corresponds to a right cyclic shift and an i-fold right cyclic shift is represented by CS^i, $1 \le i \le w$ (CS^1=CS). The S block is a $GF(2^m)$ adder which adds either all the inputs in the j-th clock cycle for $0 \le j < k-1$ or only some of them immediately prior to the last clock cycle, i.e., $j = k-1$. This is because the representation of $L_j(A, B)$

in (11) for $j = k-1$ is different from the others, i.e., when $0 \le j < k-1$. Thus, the outputs of the m AND gates at the end of S_3 are 0's just prior to the last clock cycle.

Before starting the clock (i.e., $j = 0$), the inputs of the top most block $\mathcal{F}_{w-1}$ are A and B. Thus, it generates $F_{m-1}(A,B)$ and then we have $F_{m-1}^{2^{w-1}}(A,B)$ at the output of the CS^{w-1} block. As seen in this figure, the inputs of block $\mathcal{F}_i$ are obtained from the right cyclic shift of the inputs of its upper block, i.e., $\mathcal{F}_{i+1}$. Similarly, at this time, one can verify that the outputs of $\mathcal{F}_i$, $0 \le i \le w-1$, and the CS^i blocks are $F_{m-1}(A^{2^{w-1-i}}, B^{2^{w-1-i}})$ and $F_{m-1}^{2^i}(A^{2^{w-1-i}}, B^{2^{w-1-i}})$, respectively. Thus, the inputs to the left side of the S block are the $GF(2^m)$ elements corresponding to the terms that appear in (13). Since register D is initially cleared, its w-fold cyclic shift which is input to the S block, is zero. Thus the output of the S block is $L_0(A,B)$ and it is loaded to the register D after the first clock cycle ($j = 1$).

After the first clock cycle, the contents of the register X and the register Y are A^{2^w} and $B,^{2^w}$ respectively. In general, these registers contain $A^{2^{jw}}$ and $B^{2^{jw}}$ after the j-th clock cycle. By repeated use of Lemma 3 in (11) one can see

$$\begin{aligned} L_j(A,B) &= L_{j-1}(A^{2^w}, B^{2^w}), \\ &= L_0(A^{2^{jw}}, B^{2^{jw}}), \end{aligned}$$

for $1 \le j < k-1$. This shows that after the j-th clock cycle the output of the S block is $L_j(A,B) + D^{2^w}$. It follows from (12) that after k clock cycles register D has C^{2^r}. Thus, C can be obtained by r-fold left cyclic shifts of register D. Also, one can obtain coordinates of C (not C^{2^r}) in register D by initially loading of the registers X and Y with $A^{2^{-r}}$ and $B^{2^{-r}}$ (not A and B), respectively.

Continuing from Example 2, we illustrate the proposed w-$\mathrm{SMPO_I}$ of for $w = 2$. Figure 7 depicts the structure where the F block is shown in Figure 5. It requires three clock cycles to produce the result of the multiplication.

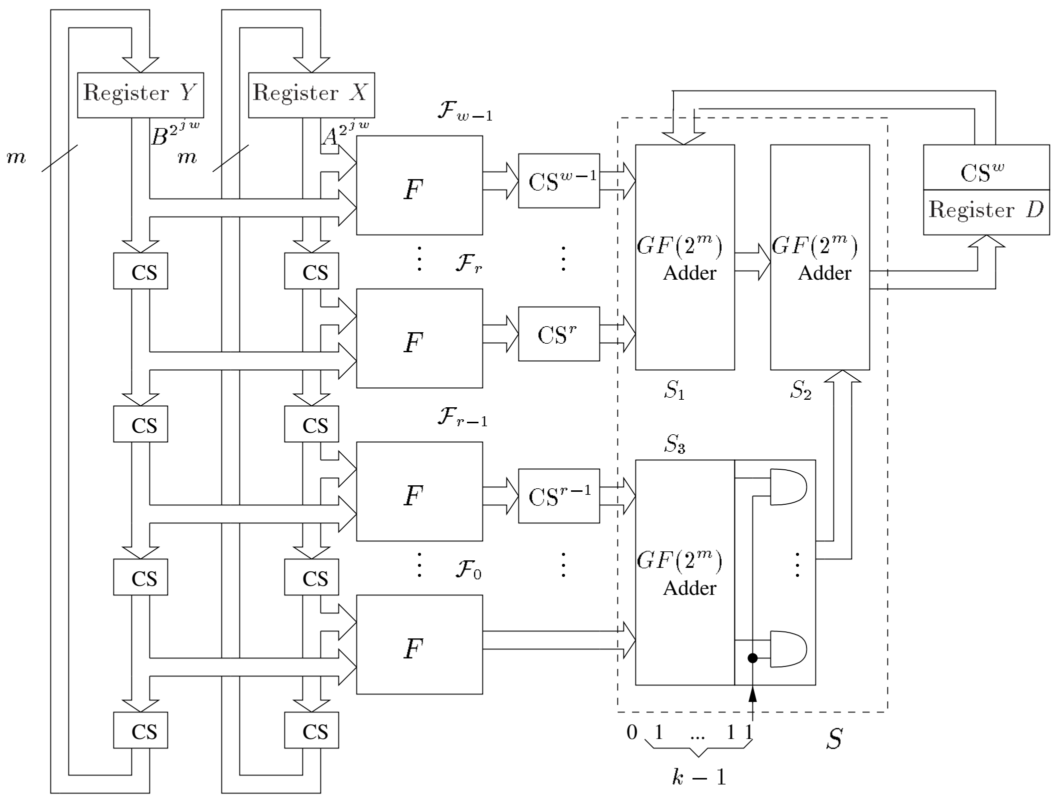

Figure 6: The architecture of word-level multiplier for arbitrary fields. CS represents the right cyclic shift and CS^i is the i-fold right cyclic shifts.

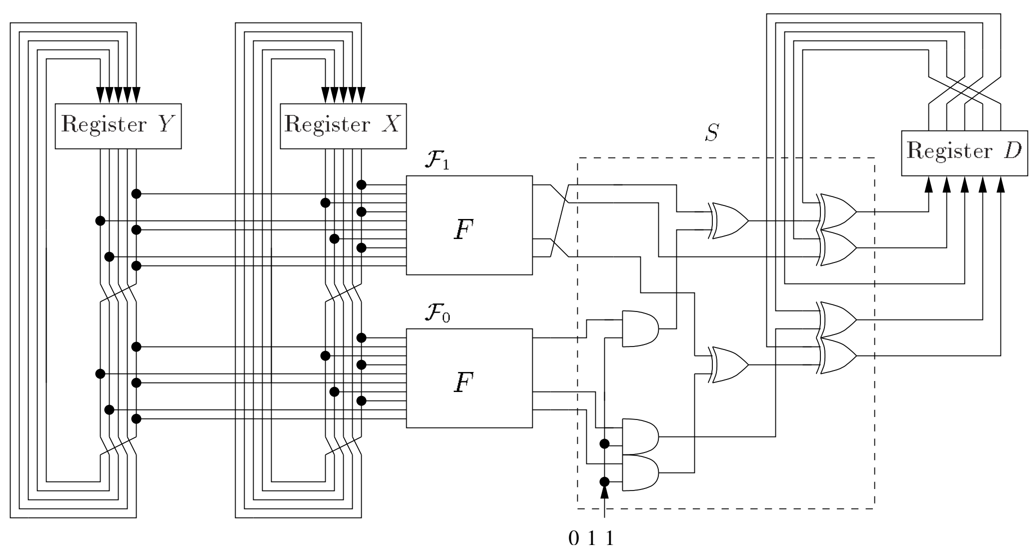

Figure 7: The proposed w-$\mathrm{SMPO}_{\mathrm{I}}$ of Example 2 for $w = 2$.

5.1 Complexities

Recall that the F block consists of the Z array, one AND gate and the XOR gates of the XOR array that are not connected to register D as seen in Figures 4(a) and 5. First we obtain the number of XOR gates in the XOR arrays of the F blocks and $GF(2^m)$ adder S by using the total number of non-zero line inputs to this part. For each individual F block the latter is $1+\sum_{j=1}^{v} H(\delta_j)$ when (5) is used. Thus, according to (9), the total number of inputs to all n XOR arrays and S block of Figure 6 is upper bounded by $m + w(1 + 0.5(C_N - 1))$. Consequently, the total number of XOR gates in the XOR arrays of F blocks and $GF(2^m)$ adder S is upper bounded by $0.5w(C_N + 1)$.

To obtain the total number of gates of the proposed word-level multiplier, we add the number of remaining gates in the F blocks and at most m AND gates in S_3 to the above value. Thus, using Table 1, one can obtain the space complexity of the word-level sequential multipliers (w-SMPO) as follows:

Proposition 2. *The gate complexities of the proposed architectures of w-SMPO$_I$ and w-SMPO$_{II}$ are*

$$\begin{aligned} \#AND &\leq w(\left\lfloor \frac{m}{2} \right\rfloor + 1) + m, \\ \#XOR &\leq \frac{w}{2}(C_N + 2m - 1), \end{aligned}$$

and

$$\begin{aligned} \#AND &\leq wm + m, \\ \#XOR &\leq \frac{w}{2}(C_N + 2\left\lfloor \frac{m}{2} \right\rfloor + 1), \end{aligned}$$

respectively.

Remark 2. *For the proposed w-SMPO$_{I/II}$, with the minimum and maximum w, i.e., $w = 1$ and $w = m$, respectively, r is zero and there is no S_3 block in Figure 6. This reduces the* *number of AND gates in Proposition 2 by m. Thus, these match with the results given in Section 3 for $w = 1$ and the best ones available in the literature, i.e., [4] and [13], for $w = m$.*

To obtain the critical delay of the proposed multipliers, first we obtain the delay of F blocks which is a T_X less than the delay of the bit-level multiplier, i.e., $T_A + \lceil \log_2 \tau \rceil T_X$. It is seen in Figure 6 that the delays of S_1, S_2 and S_2 are $\lceil \log_2 (w - r + 1) \rceil T_X$, T_X and $T_A + \lceil \log_2 r \rceil T_X$, respectively. Thus, one can state the following.

Proposition 3. *The critical delay of the proposed word-level sequential multipliers is*

$$T_A + (1 + \lceil \log_2 \tau \rceil) T_X + \max(\lceil \log_2 (w - r + 1) \rceil T_X, T_A + \lceil \log_2 r \rceil T_X). \tag{14}$$

It is noted that the last term of the critical delay given in (14) is a function of n and r, $1 \leq r \leq w-1$. Thus, the upper bound of critical delay is $2T_A + (1 + \lceil \log_2 (w - 1) \rceil + \lceil \log_2 \tau \rceil) T_X$ when $r = w - 1$.

5.2 Comparison

Table 7 compares the proposed multipliers with the word-level Massey-Omura (WLMO) multiplier, which uses w identical bit-level Massey-Omura multipliers [1], and the improved Massey-Omura (IMO) normal basis multiplier as reported in [11]. Also this table compares our proposed word-level multipliers with the recently proposed ones, namely, AND efficient digit-serial (AEDS) and XOR efficient digit-serial (XEDS) [12]. It is noted that all the previously proposed multipliers are of SMSO type which have $2m$ latches, whereas the proposed w-$\mathrm{SMPO}_{\mathrm{I/II}}$, structures are of SMPO type which need $3m$ latches. Using [12], one obtains that γ is the total number of 1's in the upper triangular matrix of $\mathbf{M}^{(1)} \vee \mathbf{M}^{(2)} \vee \cdots \vee \mathbf{M}^{(w)}$, where $\mathbf{M}^{(i)}$, $1 \leq i \leq w$, is the i-fold right and down circular shifts of all entries of the multiplication matrix $\mathbf{M}$ and $\vee$ denotes bit-wise OR operation. In general, γ is a function of w, m and normal basis N and it is not easy to obtain a closed form expression of that for an arbitrary normal basis (see [12] for details). However, one can see that $\gamma = 0.5(C_N - 1)$ for $w = 1$ and $\gamma \leq \min(\frac{w}{2}(C_N - 1),\ \frac{m}{2}(m - 1))$ for $w \leq m$.

Multipliers	#AND	#XOR	Critical Delay	Output
WLMO [1]	wC_N	$w(C_N-1)$	$T_A + \lceil \log_2 C_N \rceil T_X$	serial
IMO [11]	wm	$w(C_N-1)$	$T_A + (\lceil \log_2 \rho \rceil + \lceil \log_2 m \rceil) T_X$	serial
AEDS [12]	$\gamma + w$	$2\gamma + \frac{w}{2}(C_N-1)$	$T_A + \lceil \log_2 C_N \rceil T_X$	serial
XEDS [12]	$2\gamma + w$	$\gamma + \frac{w}{2}(C_N-1)$	$T_A + \lceil \log_2 C_N \rceil T_X$	serial
w-SMPO$_\mathrm{I}$	$w(\lfloor \frac{m}{2} \rfloor + 1) + m$	$\frac{w}{2}(C_N + 2m - 1)$	$\leq 2T_A + (1 + \lceil \log_2(w-1) \rceil + \lceil \log_2 \tau \rceil) T_X$	parallel
w-SMPO$_\mathrm{II}$	$wm + m$	$\frac{w}{2}(C_N + 2\lfloor \frac{m}{2} \rfloor + 1)$	$\leq 2T_A + (1 + \lceil \log_2(w-1) \rceil + \lceil \log_2 \tau \rceil) T_X$	parallel

Table 7: Comparison of word-level normal basis multipliers.

As seen in the table, the proposed w-SMPO$_\mathrm{I}$ and w-SMPO$_\mathrm{II}$ structures have the least time delay. Also, they have fewer number of gates than the first two structures in this table. It is difficult to compare their gate complexities with the AEDS and the XEDS structures for arbitrary normal bases. However, this can be done for optimal normal bases where $C_N = 2m-1$ and the difference between w-SMPO$_\mathrm{I}$ (w-SMPO$_\mathrm{II}$) and AEDS (XEDS) is minimum. This is shown in the following subsection.

5.3 Optimal Normal Basis

For type 1 optimal normal basis, one can simplify the architecture of Figure 6 by using the fact that $\delta_j = \beta^{2^l}$ for $1 \leq j < \frac{m}{2}$, where $2^j + 1 \equiv 2^l \mod (m+1)$, and $\delta_v = 1$, $v = \frac{m}{2}$ [4]. Thus, instead of using the normal basis N to represent the outputs of the F blocks, we use redundant basis $R = \{\beta, \beta^2, \cdots, \beta^{2^{m-1}}, 1\}$. Therefore, each F block consists of only an AND gate and a Z array whose number of Z blocks is $\frac{m}{2}$. The architecture of the sequential multiplier for type 1 optimal normal basis is shown in Figure 8. In this figure, f_i, $0 \leq i \leq w-1$, is the coordinate corresponding to "1" in R. Two binary trees of XOR gates are represented by BTX. The complexities of Figure 8 are given in Table 8. Also, the table compares the complexities of the proposed structures with those of the available word-level multipliers in the open literature. As seen in this table, our proposed structures have less time delay and about the same gate counts as reported in [12].

Remark 3. *For type 1 optimal normal bases, m is always even. Thus, if w is chosen as two, r will be zero. Therefore, the number of AND gates and the upper bound of the critical delay of the proposed w-SMPO$_{I/II}$ structures given in Table 8 are reduced by $m+1$ and a T_A*

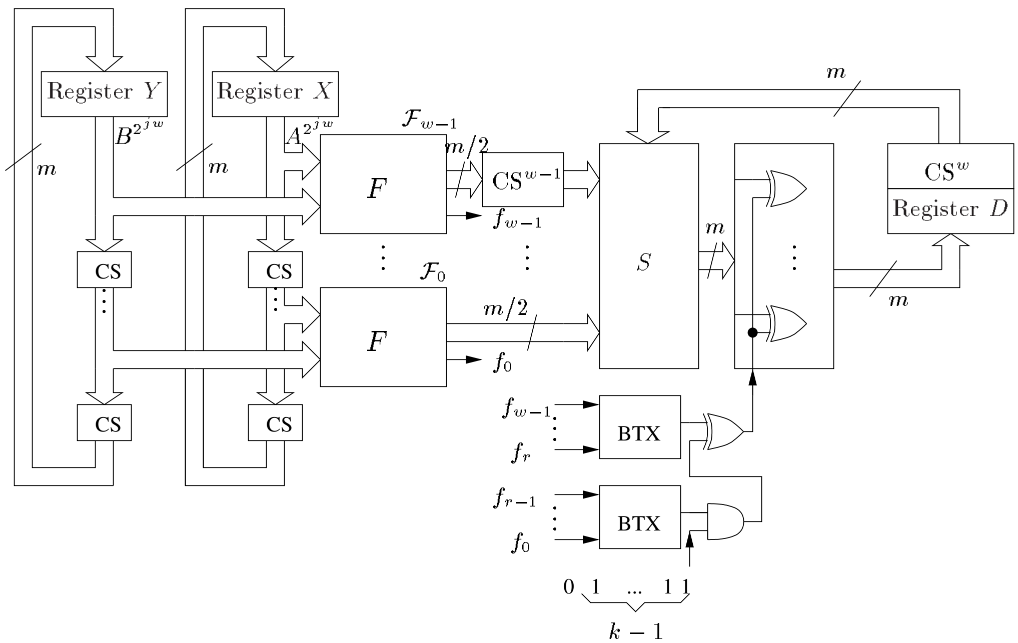

Figure 8: The architecture of word-level multiplier for type 1 optimal normal basis. BTX represents binary tree of XOR gates.

Multiplier	# AND	# XOR	Critical Delay
WLMO [1]	$w(2m-1)$	$w(2m-2)$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$
IMO[11]	wm	$w(2m-2)$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$
AEDS [12]	$(w+1)\frac{m}{2}$	$(w+1)(1.5m-2)+1$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$
XEDS [12]	$w(m-1)+m$	$(w+1)(m-1)$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$
w-SMPO$_{\mathrm{I}}$	$\frac{wm}{2}+m+w+1$	$\frac{3wm}{2}+m+w-1$	$\leq 2T_A + (3 + \lceil \log_2 (w-1) \rceil) T_X$
w-SMPO$_{\mathrm{II}}$	$wm+m+w+1$	$wm+m+w-1$	$\leq 2T_A + (3 + \lceil \log_2 (w-1) \rceil) T_X$

Table 8: Comparison among word-level sequential multipliers for type 1 optimal normal basis.

delay, respectively.

For type 2 optimal normal basis, we can directly obtain the complexity of the proposed multipliers from the general case by substituting $C_N = 2m-1$ in Propositions 2 and 3. These are compared with similar multipliers in Table 9. As seen in this table, our proposed w-SMPO$_{\mathrm{I/II}}$ structures have fewer gates and less critical delay.

Multipliers	#AND	#XOR	Critical Delay
WLMO [1]	$w(2m-1)$	$w(2m-2)$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$
IMO [11]	wm	$w(2m-2)$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$
AEDS [12]	$w(m-0.5w+0.5)$	$w(3m-w-2)$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$
XEDS [12]	$w(2m-n)$	$w(2m-0.5w-1.5)$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$
w-SMPO$_{\mathrm{I}}$	$w(\lfloor \frac{m}{2} \rfloor + 1) + m$	$w(2m-1)$	$\leq 2T_A + (3 + \lceil \log_2(w-1) \rceil)T_X$
w-SMPO$_{\mathrm{II}}$	$wm+m$	$w(m + \lfloor \frac{m}{2} \rfloor)$	$\leq 2T_A + (3 + \lceil \log_2(w-1) \rceil)T_X$

Table 9: Comparison of word-level normal basis multipliers for type 2 optimal normal basis.

6 Conclusions

In this report, we have considered multiplier architectures for $GF(2^m)$ using normal bases. For m being a multiple of an integer n, we have proposed a multiplier that uses arithmetic operations over the subfield $GF(2^n)$. We have also proposed two word-level architectures which are very useful when m is a prime or does not have a suitable divisor like n. For all the proposed multipliers, architectural level details have been presented and they have been compared with other similar multipliers in terms of number of AND gates, XOR gates, latches and critical path delays. The comparison results show that the proposed multipliers have the least number of gates and critical path delay. Such improved performance has been shown not only for arbitrary binary fields, but also for those specific binary fields that are part of the recommendation and standard of NIST and ANSI X.62 for elliptic curve digital signal algorithm.

Acknowledgments

This work has been supported in part by an NSERC postdoctoral fellowship awarded to A. Reyhani-Masoleh and in part by an NSERC grant awarded to M. A. Hasan. A preliminary version of this report was presented at the 16th IEEE Symposium on Computer Arithmetic, Santiago de Compostela, Spain, June 2003 [31].

References

[1] J. L. Massey and J. K. Omura. "Computational Method and Apparatus for Finite Field Arithmetic". *US Patent No. 4,587,627*, 1986.

[2] 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 Transactions on Computers*, 34(8):709–716, Aug. 1985.

[3] B. Sunar and C. K. Koc. "An Efficient Optimal Normal Basis Type II Multiplier". *IEEE Transactions on Computers*, 50(1):83–88, Jan. 2001.

[4] A. Reyhani-Masoleh and M. A. Hasan. "A New Construction of Massey-Omura Parallel Multiplier over $GF(2^m)$". *IEEE Transactions on Computers*, 51(5):511–520, May 2002.

[5] E. R. Berlekamp. "Bit-Serial Reed-Solomon Encoders". *IEEE Transactions on Information Theory*, 28(6):869–874, Nov. 1982.

[6] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone. "An Implementation for a Fast Public-Key Cryptosystem". *Journal of Cryptology*, 3:63–79, 1991.

[7] E. D. Mastrovito. *VLSI Architectures for Computation in Galois Fields.* PhD thesis, Linkoping Univ., Linkoping Sweden, 1991.

[8] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications.* Cambridge University Press, 1994.

[9] IEEE Std 1363-2000. "IEEE Standard Specifications for Public-Key Cryptography". January 2000.

[10] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson. "Optimal Normal Bases in $GF(p^n)$". *Discrete Applied Mathematics*, 22:149–161, 1988/89.

[11] L. Gao and G. E. Sobelman. "Improved VLSI Designs for Multiplication and Inversion in $GF(2^M)$ over normal bases". In *Proceedings of 13th Annual IEEE International ASIC/SOC Conference*, pages 97 –101, 2000.

[12] A. Reyhani-Masoleh and M. A. Hasan. "Efficient Digit-Serial Normal Basis Multipliers over $GF(2^m)$". *ACM Transactions on Embedded Computing Systems (TECS), Special issue on embedded system security*, to appear 2004.

[13] A. Reyhani-Masoleh and M. A. Hasan. "Efficient Multiplication Beyond Optimal Normal Bases". *IEEE Transactions on Computers, Special Issue on Cryptographic Hardware and Embedded Systems*, 52(4):428–439, April 2003.

[14] T. Beth and D. Gollman. "Algorithm Engineering for Public Key Algorithms". *IEEE J. Selected Areas in Communications*, 7(4):458–465, May 1989.

[15] W. Geiselmann and D. Gollmann. "Symmetry and Duality in Normal Basis Multiplication". In *LNCS-357, Proc. AAECC-6*, pages 230–238, Rome, July 1988. Springer-Verlag.

[16] M. Feng. "A VLSI Architecture for Fast Inversion in $GF(2^m)$". *IEEE Transactions on Computers*, 38(10):1383–1386, October 1989.

[17] National Institute of Standards and Technology. *Digital Signature Standard.* FIPS Publication 186-2, January 2000.

[18] D. W. Ash, I. F. Blake, and S. A. Vanstone. "Low Complexity Normal Bases". *Discrete Applied Mathematics*, 25:191–210, 1989.

[19] B. Sunar, E. Savas, and C. K. Koc. "Constructing Composite Field Representations for Efficient Conversion". *IEEE Transactions on Computers*, 52(11):1391–1398, Nov. 2003.

[20] S. Oh, C. H. Kim, J. Lim, and D. H. Cheon. "Efficient Normal Basis Multipliers in Composite Fields". *IEEE Transactions on Computers*, 49(10):1133–1138, Oct. 2000.

[21] C. Paar, P. Fleishmann, and P. Soria-Rodriguez. “Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents”. *IEEE Transactions on Computers*, 48(10):1025–1034, Oct. 1999.

[22] R. C. Mullin. “Multiple Bit Multiplier”. *US Patent No. 5,787,028*, July 1998.

[23] S. D. Galbraith and N. Smart. A Cryptographic Application of Weil Descent. In *Proceedings of the Seventh IMA Conf. on Cryptography and Coding, LNCS 1746*, pages 191–200. Springer-Verlag, 1999.

[24] N. P. Smart. How Secure Are Elliptic Curves over Composite Extension Fields? In *Proceedings of Eurocrypt 2001, LNCS 2045*, pages 30–39. Springer-Verlag, 2001.

[25] M. Maurer, A. Menezes, and E. Teske. “Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree”. *LMS Journal of Computation and Mathematics*, 5:127–174, 2002.

[26] M. Ciet, J.-J. Quisquater, and F. Sica. “A Secure Family of Composite Finite Fields Suitable for Fast Implementation of Elliptic Curve Cryptography”. In *LNCS 2247 as Proceedings of Indocrypt 2001*, pages 108–116, India, December 2001. Springer Verlag.

[27] A. Reyhani-Masoleh and M. A. Hasan. “On Low Complexity Bit Parallel Polynomial Basis Multipliers”. In *Cryptographic Hardware and Embedded Systems - CHES 2003, LNCS No. 2779, Springer Verlag, Berlin, Germany, 2003*, pages 189–202, September 2003.

[28] C. Paar. “A New Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Fields”. *IEEE Transactions on Computers*, 45(7):856–861, July 1996.

[29] American Bankers Assoc. *X9.62 American National Standards Institute Standard, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*. 1999.

[30] Chung-Chin Lu. "A Search of Minimal Key Functions for Normal Basis Multipliers". *IEEE Transactions on Computers*, 46(5):588–592, May 1997.

[31] A. Reyhani-Masoleh and M. A. Hasan. "Low Complexity Sequential Normal Basis Multipliers over $GF(2^m)$". In *Proceedings of 16th IEEE Symposium on Computer Arithmetic 2003*, pages 188–195, June 2003.