# On complexity of normal basis multiplier using modified Booth's algorithm

Jenn-Shyong Horng [1] and I-Chang Jou[1] and Chiou-Yng Lee[2]
[1] Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, R.O.C.
[2] Department of Computer Information and Network Engineering Lunghwa University of Science and Technology, Taoyuan County, Taiwan, R.O.C.
[1.] No.2 Jhuoyue Rd. Nanzih District, Kaohsiung City 811, Taiwan, R.O.C.
[2.] No. 330, Sec.1, Wanshou Rd., Guishan, Taoyuan County 33306, Taiwan, R. O. C.

*Abstract:* - This paper purposed a Booth's multiplier for normal basis multiplier. The architecture is simple and highly regular architecture for finite fields using a new modified Booth's algorithm. The proposed multiplier for finite fields requires a significantly lower number of bit level operations and, hence, can reduce the space complexity of cryptographic systems. It is shown that proposed multiplier for type-2 normal basis of GF($2^m$) saves approximately 10% space complexity as compared to related parallel multipliers. Moreover, the proposed architecture is regularity and modularity; they are well suited to VLSI implementations.

*Key-words*: - Finite field multiplication, Normal basis, Gaussian normal basis, Cryptographic.

## 1 Introduction

Efficient software algorithm and hardware implementations of arithmetic operations in the Galois field GF($2^m$) are frequently desired in coding theory, computer algebra, and elliptic curve cryptosystems [1, 2]. Of particular importance lies in the multiplication operation since it is a major building block in these applications. Other time-consuming finite field arithmetic operations such as exponentiation, division, and multiplicative inversion may be carried out by repeated multiplications. In the cryptographic system, especially in Elliptic Curve Cryptography where $m$ of GF($2^m$) field is very large. Therefore, to reduce the complexity of elliptic curve cryptosystems, efficient architectures for multiplication over GF($2^m$) are desirable.

The normal basis multiplier over GF($2^m$), as discovered by Massey and Omura [3], depends on the selection of key function for multiplication. By this way, Agnew et al.[4], Hasan et al.[5] and Sunar et al.[6] put forward hardware implementation to reduce the space complexity of the normal basis multiplier. Reyhani-Masoleh and Hasan [7, 8] also proposed a word-level normal basis multiplication architecture.

Normal basis multiplication can represent recursive shift operation of C. Each normal basis element of GF($2^m$) can also be used a Gaussian period of (m, t), called Gaussian normal basis (GNB) of GF($2^m$). The GNB is a special class of normal basis (NB) which has received considerable attention for efficient implementation of field multiplication [9]. Notably, a GNB exists only if $m$ is not divisible by eight [5]. It is to note that every GNB for GF($2^m$) is depended on an integer $t$ and is referred to as a type-t Gaussian normal basis. Reyhani-Masoleh [9] present two vector-level software algorithms for the GNB.

In this paper, the multiplier for finite field GF($2^m$) employs modified Booth's algorithm in normal basis have proposed. Booth's Algorithm [10], a uniform shift method has advantage of multiplication operation rule, and can reduce the complexity of multiplication operation. The proposed multiplier has four modules, computing A+B, triangular MUX array, computing $(\alpha Z_i)^{2^i}$ and XOR tree modules to implement GNB multiplier. By this way, the proposed NB multiplier can obtain low-complexity bit-parallel architectures.

## 2 Mathematical Background

### 2.1 Normal basis representation

It is well known that the field GF($2^m$) for all positive integer $m$ exist a normal basis (NB). Any element in $GF(2^m)$ can also be viewed as a vector space of dimension $m$ over GF($2^m$). The set $N_1 = \{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ is called a normal basis, thus the normal element $A$ in GF(2^m^) can be represented by $A = a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-1}} = (a_0, a_1, \cdots, a_{m-1})$, where $\alpha$ is normal element of GF($2^m$), if the trace function of $\alpha$ equals to one, i.e. $tr(\alpha) = 1$.

**Definition 1** Let $\gamma$ be a primitive $(mt+1)$-th root of unity in some extension field of GF(2). A Gauss period of type $(m,t)$ over GF($2^m$) is defined as

$$\alpha = \gamma + \gamma^{<2^m>_p} + \cdots + \gamma^{<2^{(t-1)m}>_p} \qquad (1)$$

where $p = mt+1$ is prime and $\gcd(tm/k, m) = 1$, where $k$ is the multiplicative order of 2 modulo $p$.

Throughout this paper, $<x>_i$ denotes the non-negative reside of $x \bmod i$. Applying the characteristics of Definition 1, $A = a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-1}}$ can also be represented as following equation:

$$\begin{aligned} A = {} & a_0(\gamma + \gamma^{<2^m>} + \cdots + \gamma^{<2^{(t-1)m}>}) \\ & + a_1(\gamma^2 + \gamma^{<2^{2m+1}>} + \cdots + \gamma^{<2^{(t-1)m+1}>}) \\ & + \cdots \\ & + a_{m-1}(\gamma^{<2^{(m-1)}>} + \gamma^{<2^{2m-1}>} + \gamma^{<2^{2m}>} + \gamma^{<2^{2m+1}>} + \cdots + \gamma^{<2^{tm-1}>}) \end{aligned}$$

Given the above element representation, normal basis $N_1$ can be rearranged by the redundant basis $N_2 = \{\gamma, \gamma^2, \cdots, \gamma^{p-1}\}$ due to $\gamma^p = 1$. By the redundant representation, normal basis $A$ may be rewritten by $A = a_{F(0)} + a_{F(1)}\gamma + a_{F(2)}\gamma^2 + \cdots + a_{F(p-1)}\gamma^{p-1}$ where $F(2^i u^j \bmod p) = i$, $u$ is an integer of order $t \bmod p$, $0 \le i \le m-1$, $0 \le j < t$ and $a_{F(0)} = 0$ or don't care item. For example, let $m=5$ and $t=2$, we use $\alpha = \gamma + \gamma^{<2^m>_p} = \gamma + \gamma^{10}$ to generate the normal basis $\{\alpha, \alpha^2, \alpha^{2^2}, \alpha^{2^3}, \alpha^{2^4}\}$. Then we obtain that $\{\gamma, \gamma^2, \gamma^3, \gamma^4, \gamma^5, \gamma^6, \gamma^7, \gamma^8, \gamma^9, \gamma^{10}\}$. By this way, $A(a_0, a_1, \cdots a_{m-1})$ can also be represented by the redundant representation

$$A = a_0\gamma + a_1\gamma^2 + a_2\gamma^4 + a_3\gamma^3 + a_4\gamma^5 + a_4\gamma^6 + a_2\gamma^7 + a_3\gamma^8 + a_1\gamma^9 + a_0\gamma^{10}.$$

Observing this representation, the coefficients of A is duplicated by t-term coefficients of the original normal basis element $A = (a_0, a_1, \cdots a_{m-1})$.

Notably, multiplication using the redundant represent is easily formed by in circular convolution structure as shown in [11]. By using the function $F(2^i u^j \bmod p) = i$, the field element $A = (a_{F(1)}, a_{F(2)}, \cdots, a_{F(p-1)})$ can be translated into the vector $(\underbrace{a_0, \cdots, a_0}_{t}, a_1, \cdots a_1, a_2, \cdots, a_2, \cdots\cdots a_{m-1}, \cdots, a_{m-1})$.

It is easily converted into the normal basis element. For example, m=5, t=2 the $F(1)=0$, $F(2)=1$, $F(3)=3$, $F(4)=2$, $F(5)=4$, $F(6)=4$, $F(7)=2$, $F(8)=3$, $F(9)=1$ and $F(10)=0$.

## 2.2 Conventional normal basis multiplication

According the above introduction the redundant basis can represent the following way. Let $A, B \in$ GF($2^m$), both elements be represented by the redundant basis representation, Thus the product $C = AB = (c_0, c_1, \cdots, c_{m-1})$ is addresses as

$$c_i = F(A,B) = \sum_{j=0}^{p-1} a_{F(j)} b_{F(i-j)} \qquad (2)$$

where $1 \le j \le p-1$ and $c_0 = 0$ (initial value)

Therefore, the product of A and B can present in Algorithm 1. This algorithm for multiplication in GF($2^m$) using a GNB of type t as described in IEEE Standard 1363-2000 [12].

**Algorithm 1** (Conventional Gaussian Normal Basis multiplication)

```
Input: $A, B \in GF(2^m)$, $F(A,B)$ in Eq.(2)
Output: $C = AB$
Step1: $C \leftarrow 0$
Step2: For $i = 0$ to $m-1$ do
    2.1 $c_i \leftarrow F(A,B)$
    2.2 $A \leftarrow A << 1$, $B \leftarrow B << 1$
Step3 return $C$
```

As mentioned above, the space complexity of NB multiplier is depended on the function $F(A,B)$. When type-1 and type-2 NB multipliers can get minimal space complexity.

## 3 Proposed NB Multiplication Algorithm

Given any elements $A, B \in$ GF($2^m$) can be represented as $A = a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-1}}$ and $B = b_0\alpha + b_1\alpha^2 + \cdots + b_{m-1}\alpha^{2^{m-1}}$, where $a_j, b_j \in$ GF(2), $0 \le j \le m-1$. For simplicity, we define the following operation algebra

**Definition 2** Let $A = a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-1}}$ be the normal element in GF($2^m$), one can be defined as $A_i = a_i\alpha + a_{i+1}\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-i-1}}$, $0 \le i \le m-1$.

Given Definition 2, the field element $A$ can be represented by extending this representation to

$$\begin{aligned} A &= a_0\alpha + a_1\alpha^2 + a_2\alpha^{2^2} + a_3\alpha^{2^3} + a_4\alpha^{2^4} + \cdots + a_{m-1}\alpha^{2^{m-1}} \\ &= a_0\alpha + (a_1\alpha + a_2\alpha^2 + a_3\alpha^{2^2} + a_4\alpha^{2^3} + \cdots + a_{m-1}\alpha^{2^{m-2}})^2 \\ &= a_0\alpha + A_1^2 \end{aligned} \quad (3)$$

Similarly,

$$\begin{aligned} B &= b_0\alpha + b_1\alpha^2 + b_2\alpha^{2^2} + b_3\alpha^{2^3} + b_4\alpha^{2^4} + \cdots + b_{m-1}\alpha^{2^{m-1}} \\ &= b_0\alpha + B_1^2 \end{aligned}$$

**Theorem 1.** According to Definition 1, $A_iB_i$ can be performed by

$$A_iB_i = \alpha Z_i + (A_{i+1}B_{i+1})^2 \quad (4)$$

Where

$$Z_i = a_iB_i + b_iA_{i+1}^2 \quad (5)$$

Proof:

Applying Definition 2, we have

$$\begin{aligned} A_i &= a_i\alpha + a_{i+1}\alpha^2 + \cdots + a_{m-1-i}\alpha^{2^{m-1-i}} \\ &= a_i\alpha + (a_{i+1}\alpha + \cdots + a_{m-1-i}\alpha^{2^{m-2-i}})^2 = a_i\alpha + A_{i+1}^2 \end{aligned}$$

Thus, $B_i$ also be represented by $B_i = b_i\alpha + B_{i+1}^2$. The product of $A_i$ and $B_i$ can be performed

$$A_iB_i = \alpha(a_iB_i + b_iA_{i+1}^2) + (A_{i+1}B_{i+1})^2 = \alpha Z_i + (A_{i+1}B_{i+1})^2 \quad ■$$

Let $AB \in \mathrm{GF}(2^m)$, by using Definition 1, A and B can be re-expressed by

$$A_0 = a_0\alpha + A_1^2$$

$$B = b_0\alpha + B_1^2$$

Thus, the product C of A and B can obtain the following formulae

$$\begin{aligned} C &= AB = (a_0\alpha + A_1^2)(b_0\alpha + B_1^2) \\ &= a_0\alpha(b_0\alpha + B_1^2) + b_0\alpha A_1^2 + (A_1B_1)^2 \\ &= \cdots \\ &= \alpha(a_0B_0 + b_0A_1^2) + (\alpha(a_1B_1 + b_1A_2^2) + \cdots + (A_{m-1}B_{m-1})^2)^2 \cdots)^2 \\ &= \alpha(a_0B_0 + b_0A_1^2) + (\alpha(a_1B_1 + b_1A_2^2))^2 + \cdots + (\alpha(a_{m-1}B_{m-1} + b_{m-1}A_m^2))^{2^{m-1}} \\ &= \alpha Z_0 + (\alpha Z_1)^2 + (\alpha Z_2)^{2^2} + (\alpha Z_3)^{2^3} + \cdots + (\alpha Z_{m-1})^{2^{m-1}} \end{aligned} \quad (6)$$

or

$$= (\cdots((\alpha Z_{m-1})^2 + \alpha Z_{m-2})^2 + \alpha Z_{m-3})^2 \cdots)^2 + \alpha Z_0 \quad (7)$$

As stated above, the product C is lied on $\alpha Z_i$. For efficient solving the $\alpha Z_i$ operation will proposed following steps:

Step 1. Convert $Z_i$ to redundant basis

$$Z_i = z_{i,0}\gamma + z_{i,1}\gamma^2 + z_{i,2}\gamma^3 \cdots + z_{i,p-1-i}\gamma^{p-1-i}$$

Step 2. Calculate

$$\gamma\, Z_i = \gamma(z_{i,0}\gamma + z_{i,1}\gamma^2 + z_{i,2}\gamma^3 \cdots + z_{i,p-1-i}\gamma^{p-1-i})$$

Because $\gamma^p = 1$ then

$$\begin{aligned} \gamma\, Z_{\mathrm{i}} &= z_{i,0}\gamma^2 + z_{i,1}\gamma^3 + z_{i,2}\gamma^4 \cdots + z_{i,p-1-i}\gamma^{p-i} \\ &= Z_i^{(1)} \end{aligned}$$

is the one-fold right shift of the binary vector representation of $Z_i$.

Similarly, $\gamma^{<2^k>_p} Z_{\mathrm{i}} = Z_i^{(<2^k>_p)}$.

So $\alpha$ and $Z_i$ product obtained

$$\begin{aligned} \alpha Z_{\mathrm{i}} &= (\gamma + \gamma^{<2^m>_p} + \cdots + \gamma^{<2^{m(t-1)}>_p})Z_i \\ &= Z_i^{(1)} + Z_i^{(<2^m>_p)} + \cdots + Z_i^{(<2^{m(t-1)}>_p)} \\ &= \sum_{j=0}^{p-1} z_{i,F(<j-1>_p)}\gamma^j + \sum_{j=0}^{p-1} z_{i,F(<j-<2^m>_p>_p)}\gamma^j + \cdots + \sum_{j=0}^{p-1} z_{i,F(<j-<2^{m(t-1)}>_p>_p)}\gamma^j \\ &= \sum_{j=0}^{p-1} z'_{i,F(j)}\gamma^j = z'_{i,F(0)} + z_{i,0}\alpha + z_{i,1}\alpha^2 + z_{i,2}\alpha^{2^2} + \cdots + z_{i,m-1-i}\alpha^{2^{m-1-i}} \end{aligned}$$

where

$$\begin{aligned} & z_{i,F(<1>_p)} + z_{i,F(<2^m>_p)} + \cdots + z_{i,F(<2^{m(t-1)}>_p)} = z'_{i,F(0)} \\ & z_{i,F(<0>_p)} + z_{i,F(<1-2^m>_p)} + \cdots + z_{i,F(<1-2^{m(t-1)}>_p)} = z_{i,0} \\ & \ldots \\ & z_{i,F(<j>_p)} + z_{i,F(<j+1-2^m>_p)} + \cdots + z_{i,F(<j+1-2^{m(t-1)}>_p)} = z_{i,j} \end{aligned} \quad (8)$$

when $t$ is even (then $m$ is odd), the Eq.(8) can obtain

$$z_{i,F(<1>_p)} + z_{i,F(<2^m>_p)} + \cdots + z_{i,F(<2^{m(t-1)}>_p)} = 0$$

when $t$ is odd (then $m$ is even) , the Eq.(8) can obtain

$$z_{i,F(<1>_p)} + z_{i,F(<2^m>_p)} + \cdots + z_{i,F(<2^{m(t-1)}>_p)} = z_{i,m/2}$$

Step 3. According to Eq.(8), the $\alpha Z_i$ can easily be converted into normal basis.

**Example:** For type 2 GNB over $\mathrm{GF}(2^5)$, then the $\alpha Z_i$ obtain following $p = mt + 1 = 11$ and $\alpha = \gamma + \gamma^{2^5}$, then

$$\begin{aligned} \alpha Z_i &= (\gamma + \gamma^{<2^5>_{11}})Z_i \\ &= Z_i^{(1)} + Z_i^{(10)} \\ &= z'_{i,F(0)} + z'_{i,F(1)}\gamma + z'_{i,F(2)}\gamma^2 + \cdots + z'_{i,F(9)}\gamma^9 + z'_{i,F(10)}\gamma^{10} \\ &= z'_{i,F(0)} + z'_{i,0}\alpha + z'_{i,1}\alpha^2 + z'_{i,2}\alpha^{2^2} + z'_{i,3}\alpha^{2^3} + z'_{i,4}\alpha^{2^4} \end{aligned}$$

and $z'_{i,F(0)} = 0$ , $z'_{i,0} = z_{i,0} + z_{i,1}$ , $z'_{i,1} = z_{i,0} + z_{i,3}$ , $z'_{i,2} = z_{i,3} + z_{i,4}$ , $z'_{i,3} = z_{i,1} + z_{i,2}$ , $z'_{i,4} = z_{i,2} + z_{i,4}$ , then can obtain $\alpha Z_i$ is normal basis.

As mentioned Definition 2 and Theorem 1, we can derive out Eq.(6), and NB multiplication algorithm then is derived as follows.

**Algorithm 2**

Input: $A = a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-1}}$ , $B = b_0\alpha + b_1\alpha^2 + \cdots + b_{m-1}\alpha^{2^{m-1}}$ ,

```
$F(\alpha)=\alpha C \quad A,B \in GF(2^m), a_i,b_i \in GF(2)$
Output: $C = AB$
Step 1 $C \leftarrow 0$
    $D = A + B$
Step 2 For $i=0$ to $m-1$ do
    2.1 If $(a_i == 1 \text{ and } b_i == 1)$
    2.2 then $C = D + C$
    2.3 else If $(a_i == 1 \text{ and } b_i == 0)$
    2.4 then $C = A + C$
    2.5 else If $(a_i == 0 \text{ and } b_i == 1)$
    2.6 then $C = B + C$
    2.7 $C = \alpha C$
    2.8 $C = C << 1$
Step 3 Return $C$
```

**Theorem 2** Calculate $(\alpha Z_i)^{2^i}$ sum need one XOR gate time delay and space complexity less than $mt$ XOR gates.

**Theorem 3** For the type 2 and 1 GNB over $\mathrm{GF}(2^m)$, the XOR tree is need one XOR time delay and $m-1$ XOR gates space complexity.

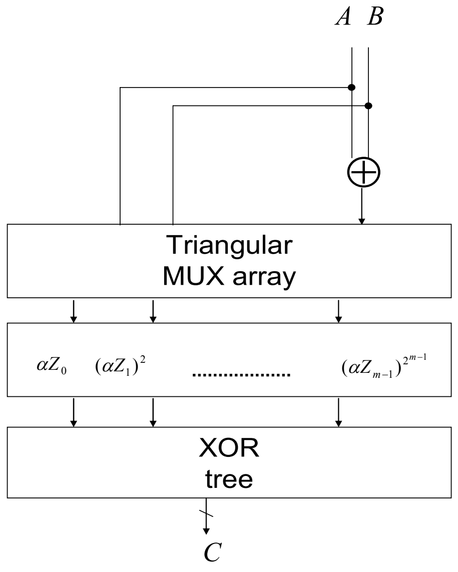
## 4 Hardware architecture and complexity analysis

As aforementioned, the Eq.(5) $Z_i = a_i B_i + b_i A_{i+1}^2$ function using the Table 1 to invest with function decision table, and then, implement the function architecture as Figure 1. In Table 1 MUX function $a_i$ and $b_i$ are select parameter and the $Z_i = a_i B_i + b_i A_{i+1}^2$ column is output function. Hence, using the Table 1, select parameter and output function can implement to Figure 1. As a result, using Eq.(5) and Eq.(6) implement in Figure 2 shown the proposed NB multiplier, which it includes computing unit, triangular MUX array unit, computing $(\alpha Z_i)^{2^i}$ and XOR tree unit. Therefore, the multiplier elements uses AND gate, XOR gate and MUX unit to carry out the normal basis multiplier. For constructing the computing unit, it using XOR gate computing $(\alpha Z_i)^{2^i}$. And in the triangular MUX array unit, it including upper triangular MUX array, as shown in Figure 2. Finally, in the XOR tree unit, it including XOR tree, as shown in Figure 3. Figure 2 is shown through the MUX unit computing get the $(\alpha Z_i)^{2^i}$. Further, implement in Figure 3 is shown have the parameters $(\alpha Z_i)^{2^i}$ into the XOR tree.

The following became clear after the investigation on the $Z_i = a_i B_i + b_i A_{i+1}^2$ function decision table.

**Table 1:** $Z_i = a_i B_i + b_i A_{i+1}^2$ function decision table

| $a_i$ | $b_i$ | $(\alpha Z_i)^{2^i}$ |
|---|---|---|
| **0** | **0** | **0** |
| **0** | **1** | $(\alpha A_{i+1}^2)^{2^i}$ |
| **1** | **0** | $(\alpha B_i)^{2^i}$ |
| **1** | **1** | $(\alpha(B_i + A_{i+1}^2))^{2^i}$ |



A B
Triangular MUX array
$\alpha Z_0$ $(\alpha Z_1)^2$ .................. $(\alpha Z_{m-1})^{2^{m-1}}$
XOR tree
C

**Figure 1:** normal basis multiplier architecture

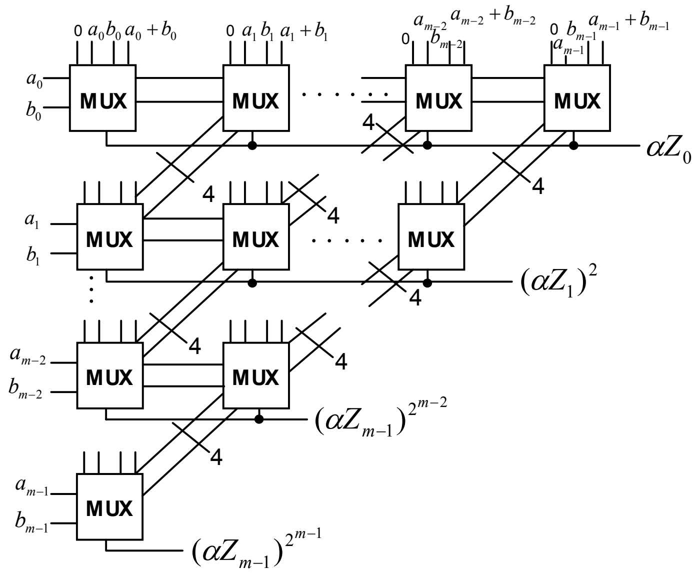


**Figure 2:** computing the $(\alpha Z_i)^{2^i}$ over GF($2^m$)

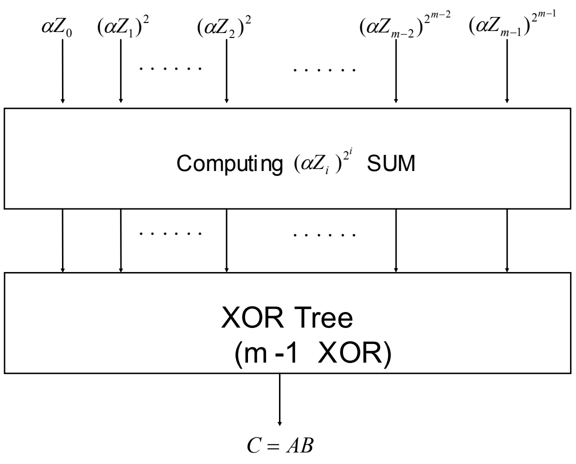

$\alpha Z_0$
$(\alpha Z_1)^2$
$(\alpha Z_2)^2$
$(\alpha Z_{m-2})^{2^{m-2}}$
$(\alpha Z_{m-1})^{2^{m-1}}$
Computing $(\alpha Z_i)^{2^i}$ SUM
XOR Tree (m -1 XOR)
$C = AB$

**Figure 3:** normal basis multiplier computing $(\alpha Z_i)^{2^i}$ sum and through XOR tree over GF($2^m$)

Synthesize above-mentioned two kinds of multipliers and compare its complexity, architecture to implement this multiplication is shown in Figure 1. Its architecture includes the computing $(\alpha Z_i)^{2^i}$ unit, triangular MUX array and XOR tree. The computing $(\alpha Z_i)^{2^i}$ unit needs XOR gates. In Figure 2, the triangular MUX array includes four inputs and one output MUX unit needs the total number is $m(m-1)/2$ units. In Figure 3, XOR tree needs *(m-1)* XOR gates. The total size complexity is $m^2+2m-1$ XOR gates, $m$ AND gates and $m(m-1)/2$ MUX units.

The computing path delay of the multiplier includes the computing $Z_1$ block ($T_X$), the MUX array ($T_{MUX}$), and the XOR tree ($[\log_2(m-1)]T_X$). Therefore, the total delay is $2T_X + T_{MUX} + (\log_2 m - 1)T_X$. From the facts described above, we may show that in Table 3.

More precisely, the transistors are counted using a standard CMOS VLSI. In the CMOS VLSI technology, 2-input XOR, 2-input AND , 4x1 MUX and 1-bit latch comprise 6, 6, 16 and 8 transistors, respectively [14]. Table 3 compares various multipliers with transistor counts. The results demonstrate that the proposed multiplier saves approximately 10% space complexity as compared to usual existing bit-parallel multipliers [6, 15]. As stated above, the proposed architecture requires very small transistor counts, bet the latency as compared to the multipliers is approximately for [3, 6, 15]. It is noted that NIST recommended five binary fields for the ECDSA (Elliptic Curve Digital Signature Algorithm) applications. These are GF($2^{163}$), GF($2^{233}$), GF($2^{283}$), GF($2^{409}$) and GF($2^{571}$), but no irreducible trinomials and AOPs exist for three degrees, viz., 163, 283 and 571 [16]. Accordingly, applications of these circuits for trinomials and AOPs are then limited. The proposed multiplier does not encounter such problems.

**Table 2:** proposed type-II multiplier space complexity and time delay

| | Space complexity | | | Time delay |
|---|---|---|---|---|
| | #AND | #XOR | #MUX | |
| computing $A+B$ | $m$ | $m-1$ | -- | $T_X$ |
| MUX array | -- | -- | $\frac{m(m-1)}{2}$ | $T_{MUX}$ |
| computing $(\alpha Z_i)^{2^i}$ | -- | $\le 2m$ | -- | $T_X$ |
| XOR tree | -- | $m(m-1)$ | -- | $[\log_2(m-1)]T_X$ |
| total | $m$ | $m^2+2m-1$ | $\frac{m(m-1)}{2}$ | $T_{MUX}+[2+\log_2(m-1)]T_X$ |

**Table 3:** Comparison of Multipliers When There is a Type II ONB

| Multipliers | Model | # AND | # XOR | # Flip Flops or # MUX | Time delay |
|---|---|---|---|---|---|
| Agnew et al. [15] | Bit-Par allel | $m^2$ | $\frac{3m(m-1)}{2}$ | -- | $T_A+[1+\log_2(m-1)]T_X$ |
| Sunar et al. [17] | Bit-Par allel | $m^2$ | $\frac{3m(m-1)}{2}$ | -- | $T_A+[1+\log_2(m-1)]T_X$ |
| proposed | Bit-Par allel | $m$ | $m^2+2m-1$ | $\frac{m(m-1)}{2}$ | $T_{MUX}+[2+\log_2(m-1)]T_X$ |

## 5 Conclusions

The main result of this study is that proposed the new NB multiplier employs modified Booth's algorithm has low space complexity architecture. In Table 3, it is demonstrated that the proposed multiplier has lower hardware complexity than other existing multipliers. Therefore, this algorithm is suitable for implementation of cryptographic function in software and hardware. Furthermore, the study of normal basis multiplier is need being continued steadily.

From the facts described above, we may conclude that this makes the architecture very useful in finite field applications, especially in Elliptic Curve Cryptography where $m$ of GF($2^m$) field is very large. Therefore, the presented architecture is appropriate for special applications with particular limited space constraints such as smart cards, mobile phones or other portable devices.

References

[1] F. J. Macwilliams and N. J. A. Sloane, The theory of error-correcting codes, *Amsterdam: North-Holland*, 1977.

[2] R. Lidl and H. Niederreiter, Introduction to finite fields and their applications, *New York: Cambridge Univ. Press*, 1994.

[3] J. L. Massey and J. K. Omura, “Computational Method and Apparatus for Finite Field Arithmetic,” US Patent No. 4587627, 1986.

[4] G. B. Agnew, R. C. Mullin, I M. Onyszchuk, and S. A. Vanstone, “An Implementation for a Fast Public-Key Cryptosystem,” *Journal of Cryptology*, vol.3, pp. 63-79, 1991.

[5] M. A. Hasan, M. Z. Wang, and V. K. Bhargava, “A Modify Massey-Omura Parallel Multiplier for a Class of Finite Fields,” *IEEE Transactions on Computers*, vol. 42, no. 10, pp. 1278-1280, Oct. 1993.

[6] D. W. Ash, I. F. Blake, and S. A. Vanstone, “Low Complexity Normal basis,” *Discrete Applied Mathematics*, 25:191-210, 1989.

[7] A. Reyhani-Masoleh and M. A. Hasan, “Fast Normal Basis Multiplication Using General Purpose Processors,” *IEEE Transactions on Computers*, vol. 52, no. 11, pp. 1379-1390, Nov. 2003.

[8] A. Reyhani-Masoleh and M. A. Hasan, “Low Complexity Word-Level Sequential Normal Basis Multiplier,” *IEEE Transactions on Computers*, vol. 54, no. 2, pp. 98-110, Feb. 2005.

[9] A. Reyhani-Masoleh, “Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases,” *IEEE Transactions on Computers*, vol. 55, no. 1, pp. 34-47, Jan. 2006.

[10] A. D. Booth, “A signed binary multiplication technique,” Quart. J. Mech, Appl. Math., Vol. 4, Part 2, 1951.

[11] C. Y. Lee, E. H. Lu, and J. Y. Lee, “Bit-parallel systolic multipliers for GF($2^m$) fields defined by all-one and equally-spaced polynomials,” *IEEE Transactions on Computers*, vol. 50, no. 5, pp. 385-393, May 2001.

[12] IEEE Standard 1363-2000, “Standard Specifications for Public Key Cryptography,” Aug. 2000.

[13] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson, “Optimal Normal Basis in GF($p^n$),” Discrete Applied Mathematics, Vol. 22, pp. 149-161, 1988/89.

[14] K. Z. Pekmestzi, “Multiplexer-based array multipliers,” *IEEE Transactions on Computers* , vol. 48, no. 1, pp. 15-23, Jan. 1999.

[15] A. Reyhani-Masoleh and M. A. Hasan, “ A New Construction of Massey-Omura Parallel Multiplier over GF($2^m$),” *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 511-520, May. 2002.

[16] G. Seroussi, “Table of low-weight binary irreducible polynomials,” *Technical Report HPL-98-135, Hewlett-Packard Laboratories, Palo Alto, Calif.*, Aug. 1998, Available at http://www.hpl.hp.com/techreports/98/HPL-98-135.html.

[17] B. Sunar and C. K. Koc, “An Efficient Optimal Normal Basis Type II Multiplier,” *IEEE Transactions on Computers*, vol. 50, no. 1, pp. 83-88, Jan. 2001.