An Efficient Reconfigurable II-ONB Modular Multiplier

Li Miao[1,2], He Liangsheng[1], Yang Tongjie[1,*] Gao Neng[2], Liu Zongbin[2], and Zhang Qinglong[2]

[1] Zhengzhou Information Science and Technology Institute, Zhengzhou, 450004, China
[2] SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China
{limiao12,lshhe,tjyang,gaoneng,zbliu,qlzhang}@is.ac.cn

Abstract. In Elliptic Curve Cryptography(ECC), due to the characteristic of high efficiency, the modular multiplication operation in type II optimal normal basis(II-ONB) over binary field has become a key research trend. Based on B. Sunar's basis conversion theory, in this paper, an improved II-ONB modular multiplication algorithm has been proposed and an efficient reconfigurable modular multiplier, which can support different lengths has been implemented. This work has been simulated using ModelSim and synthesized under 0.18μm CMOS technology. Then, complexity comparison has also been accomplished. The results prove that our proposed reconfigurable II-ONB modular multiplier can not only guarantee high flexibility for arbitrary modular multiplication, but also have area advantage in resource-constrained ECC applications.

Keywords: Elliptic Curve Cryptography; Type II Optimal Normal Basis; Reconfigurable Modular Multiplier; Basis Conversion.

1 Introduction

In modern age, along with flourishing development of E-Commerce, E-Administration and military communications, information security has been more and more widely focused by people. The public-key cryptography system can effectively solve problems like anti-repudiation, authentication and key distribution on public channels. Based on the elliptic curve discrete logarithm problem(ECDLP), Elliptic Curve Cryptography(ECC)[1-2] has already been proved to be more secure and more efficient than RSA. Therefore, ECC has gradually replaced RSA as the next generation of public-key cryptography standard[3]. Modern cryptanalysis indicates that ECC provides high security strength per bit[4], so at the same security level, it can offer the fastest computation, the least storage requirement and the lowest communication bandwidth, which is very suitable for resource restriction devices[5-6] like mobile telephones, PDA, wireless network and smart cards. In fields of high-end applications, such as network server and certificate authority, due to large secure connection number and certain real-time requirement, ECC can provide signature authentication service with higher throughput, too.

[*] Corresponding author.

Finite field multiplication is a critical operation for ECC implementation performance, and how to design an efficient and flexible finite field multiplier has also become a focus in cryptographic applications[7-11]. At present, these are two main implementation methods for large integer multiplication: the software and the hardware. The software method is highly flexible and convenient to use, but restricted by the general purpose microprocessor instruction system, the operation efficiency is so low that it can't meet the need for high speed applications. Therefore, it's necessary to design application specific integrated circuit(ASIC). This hardware method can reach high speed and low power consumption, but the specific property in structure is too inflexible to further development. ECC usually chooses keys with different lengths for information graded protection. When key length changes, the hardware circuit must be redesigned that results in a waste of manpower and material resources. In the meanwhile, it increases ECC chip types and managing difficulty. One efficient solution is to design a kind of parameter reconfigurable hardware circuit to improve ECC chips flexibility, in which the reconfigurable design of finite field multiplier is the kernel[12-14].

Normal basis[15] is one of the most important representation over binary field. Currently, there is no flexible and reconfigurable design scheme for normal basis multiplier at all times. In order to simplify extremely complicated multiplication operation in normal basis, researchers have found a special class of normal basis called optimal normal basis(short for ONB)[16]. The ONB has the lowest computational complexity, whose exponentiation and multiplication operations are very simple. Type I ONB and type II ONB(short for II-ONB) are two kinds of most commonly used ONB[17], while with the highest efficiency, II-ONB multiplication operation has been widely used. For $GF(2^m)$, there are 174 m values in the range $m\in[2,1000]$, for which a II-ONB exists. And multiplication matrix M of II-ONB has the minimum number of "1", which is equal to $2m$-1. Except the first column, every other column has only two "1", which greatly reduces space complexity and computational complexity of modular multiplication operation. Consequently, designs for II-ONB multiplier have become hot.

In 2001, B.Sunar proposed an idea and concrete method of basis conversion[18], which provided a new thinking for II-ONB multiplier. In 2008, A. H. Namin of Canada Windsor University brought forward a word-level multiplier for II-ONB[19]. This multiplier had advantage in computation speed, but disadvantage in circuit area. Moreover, it could not support modular multiplication operation with variable lengths. In 2009, T. F. Al-Somani of Saudi Umm Kula University presented an improved Massey-Omura normal basis multiplier using three-stage pipelines[20]. It had been advanced significantly in performance, but could not support modular multiplication operation with diverse lengths, too.

Up to now, almost all II-ONB multipliers were designed fixed in structure and only achieved a sort of specific ECC operation over binary field. The bad flexibility was difficult to meet the need for ECC flexible processing. Therefore, this paper aims to do some research and design an efficient reconfigurable II-ONB multiplier to meet the needs of II-ONB multiplication operation with different lengths, and provide a new design method and technology for solving problems of II-ONB multiplier in multiplication operation with single length and poor flexibility.

2 Type II Optimal Normal Basis and Multiplication Operation

For element $\beta \in GF(2^m)$, a normal basis can be expressed as $\{\beta, \beta^2, \cdots, \beta^{2^{m-1}}\}$ and the corresponding normal polynomial is defined as $F(t)=t^m + c_{m-1}t^{m-1} + \cdots + c_1 t + c_0$. II-ONB can be constructed[17] if $2m+1$ is a prime and if either 2 is a primitive root modulo $2m+1$ or $2m+1=3 \pmod 4$ and the multiplicative order of 2 modulo $2m+1$ is m holds. Then, $\alpha = r + r^{-1}$ generates an optimal normal basis for $GF(2^m)$, where r is a primitive $(2m+1)$th root of unity, i.e., $r^{2m+1}=1$ and $r^i \neq 1$ for any $1 \le i < 2m+1$.

In normal basis, the square operation of A is just simple cyclic shift operation, namely $A^2=(a_m\ a_1\ a_2 \cdots a_{m-1})$. However, the multiplication operation is relatively complex. Firstly, a multiplication matrix M should be computed, whose calculation steps are as follows[21]:

1. Calculate the convert matrix E from $(1, t, \cdots, t^{m-1})$ to $(t, t^2, \cdots, t^{2^{m-1}})$:

$$\begin{aligned}
t &= e_{0,0} + e_{0,1}\cdot t + e_{0,2}\cdot t^2 + \cdots + e_{0,m-1}\cdot t^{m-1} \bmod F(t) \\
t^2 &= e_{1,0} + e_{1,1}\cdot t + e_{1,2}\cdot t^2 + \cdots + e_{1,m-1}\cdot t^{m-1} \bmod F(t) \\
t^4 &= e_{2,0} + e_{2,1}\cdot t + e_{2,2}\cdot t^2 + \cdots + e_{2,m-1}\cdot t^{m-1} \bmod F(t) \\
&\cdots \\
t^{2^{m-1}} &= e_{m-1,0} + e_{m-1,1}\cdot t + e_{m-1,2}\cdot t^2 + \cdots + e_{m-1,m-1}\cdot t^{m-1} \bmod F(t)
\end{aligned}$$

The coefficients meeting all above equations form convert matrix E:

$$E = \begin{bmatrix} e_{0,0} & e_{0,1} & \cdots & e_{0,m-1} \\ e_{1,0} & e_{1,1} & \cdots & e_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ e_{m-1,0} & e_{m-1,1} & \cdots & e_{m-1,m-1} \end{bmatrix}$$

2. Calculate the inverse matrix of E: $G=E^{-1}$.
3. Suppose matrix Q is expressed as:

$$Q = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ q_0 & q_1 & q_2 & \cdots & q_{m-1} \end{bmatrix}$$

Calculate matrix $D = EQG$.

4. Suppose $\mu_{i,j}=d_{j-i,-i}(i, j=0, 1, \cdots, m-1)$ and subscripts of d get the least non-negative integer values of modulo m. The multiplication matrix M is obtained:

$$M = \begin{bmatrix} \mu_{0,0} & \mu_{0,1} & \cdots & \mu_{0,m-1} \\ \mu_{1,0} & \mu_{1,1} & \cdots & \mu_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{m-1,0} & \mu_{m-1,1} & \cdots & \mu_{m-1,m-1} \end{bmatrix}$$

The product of $A \cdot B$ in normal basis can be expressed as $C{=}AMB^T$, where B^T denotes the transpose of B.

The modular multiplication algorithm in normal basis is shown as Algorithm 1. The core operation is the matrix multiplication (step 3), which calculates 1 bit product in every cycle.

Algorithm 1. Modular multiplication algorithm in normal basis[21]

```
Input: $A = (a_1, a_2, \cdots, a_m), B = (b_1, b_2, \cdots, b_m)$,
        $multiplication\ matrix\ M\ for\ GF(2^m)$
Output: Product $C = (c_1, c_2, \cdots, c_m)$
1. $x = A, y = B$
2. for $i = 1$ to $m$ do
3.     $c_i = f(x, y) = xMy^T$
4.     $x <<< 1$  $(1\ bit\ cyclic\ left\ shift\ of\ x)$
5.     $y <<< 1$  $(1\ bit\ cyclic\ left\ shift\ of\ y)$
   end for
6. $C = (c_1, c_2, \cdots, c_m)$
```

3 II-ONB Modular Multiplication Algorithm

There are two important steps during the design of II-ONB modular multiplier: the first step is converting elements represented in II-ONB to a specific basis $\underline{N}'$, which makes multiplication operation in basis $\underline{N}'$ have a regular representation; the second step is multiplying the elements in basis $\underline{N}'$.

3.1 Basis Conversion Theory

According to II-ONB construction method, for $GF(2^m)$, normal element $\beta = r + r^{-1}$, where r is a primitive $(2m + 1)$th root of unity, i.e., $r^{2m+1} = 1$ and $r^i \neq 1$ for any $i \in [1, 2m{+}1)$, the normal basis is given as $\underline{N} = \{\beta, \beta^2, \cdots, \beta^{2^{m-1}}\}$. If 2 is a primitive root modulo $2m{+}1$, then the set of powers of 2 modulo $2m + 1$ is:

$$P_1 = \{2, 2^2, \cdots, 2^{2m}\} \bmod (2m+1) \tag{1}$$

Equation (1) is equivalent to $Q_1 = \{1, 2, \cdots, 2m\}$. Therefore, a basis element $r^{2^i} + r^{-2^i}$ can be written as $r^j + r^{-j}$ for $j \in [1, 2m]$. Furthermore, it is always possible to rewrite $r^j + r^{-j}$ as $r^{(2m+1)-j} + r^{-(2m+1)+j}$. If $j \geqslant m{+}1$, then this has the benefit of bringing the power of r to the range $[1, m]$. If the multiplicative order of 2 modulo $2m{+}1$ is equal to m, then the set of powers of 2 modulo $2m + 1$ is:

$$P_2 = \{2, 2^2, \cdots, 2^m\} \bmod (2m+1) \tag{2}$$

Equation (2) is equivalent to $Q_2 = \{1, 2, \cdots, m\}$. As a result, a basis element $r^{2^i} + r^{-2^i}$ can be written uniquely as $r^j + r^{-j}$ with $j \in [1, m]$. The basis whose element is $r^j + r^{-j}$ is defined as $\underline{N}' = \{\beta_1, \beta_2, \cdots, \beta_m\}$, where $\beta_j = r^j + r^{-j}$ and $j \in [1, m]$.

Let A be expressed in the basis $\underline{N}$ as $A = a_1'\beta + a_2'\beta^2 + a_3'\beta^4 + \cdots + a_m'\beta^{2^{m-1}}$ where $\beta = r + r^{-1}$. The representation of A in the basis $\underline{N}'$ is given as $A = a_1\beta_1 + a_2\beta_2 + a_3\beta_3 + \cdots + a_m\beta_m$ where $\beta_i = r^i + r^{-i}$. We can express the permutation between the coefficients $a_j = a_i'$ as[18,22]:

$$j = \begin{cases} k & if \ \ k \in [1, m] \\ (2m+1) - k & if \ \ k \in [m+1, 2m] \end{cases} \tag{3}$$

Where $k = 2^{i-1} (\bmod 2m + 1)$ for $i = 1, 2, \cdots, m$. Not a normal basis, basis $\underline{N}'$ is just a shifted form of canonical basis[16].

3.2 II-ONB Modular Multiplication Algorithm Based on Basis Conversion

Adopt Equation (3) for basis conversion. $A, B \in GF(2^m)$, can be represented in basis $\underline{N}'$ as $A = \sum_{i=1}^{m} a_i\beta_i = \sum_{i=1}^{m} a_i(r^i + r^{-i})$ and $B = \sum_{i=1}^{m} b_i\beta_i = \sum_{i=1}^{m} b_i(r^i + r^{-i})$. Then product $C = A \cdot B$ can be written as:

$$\begin{aligned} C = A \cdot B &= \left(\sum_{i=1}^{m} a_i(r^i + r^{-i})\right)\left(\sum_{j=1}^{m} b_j(r^j + r^{-j})\right) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} a_ib_j(r^{i-j} + r^{-(i-j)}) + \sum_{i=1}^{m}\sum_{j=1}^{m} a_ib_j(r^{i+j} + r^{-(i+j)}) \\ &= C_1 + C_2 \end{aligned} \tag{4}$$

If $i = j$, then $r^{i-j} + r^{-(i-j)} = r^0 + r^0 = 0$, so the coefficients of β_k in C_1 are the sum of all a_ib_j for which $k = |i - j| \in [1, m]$. In C_2, $|i + j| \in [1, 2m]$ can be divided into $|i + j| \in [1, m]$ and $|i + j| \in [m+1, 2m]$, while the latter case can be replaced by $2m + 1 - |i + j|$. So C_2 can be transformed into the following:

$$\begin{aligned} C_2 &= \sum_{i=1}^{m}\sum_{j=1}^{m} a_ib_j(r^{i+j} + r^{-(i+j)}) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m-i} a_ib_j(r^{i+j} + r^{-(i+j)}) + \sum_{i=1}^{m}\sum_{j=m-i+1}^{m} a_ib_j(r^{i+j} + r^{-(i+j)}) \\ &= D_1 + D_2 \end{aligned} \tag{5}$$

In Equation (5), the coefficients of β_k in D_1 are the sum of all a_ib_j for which $k=|i+j|\in[1, m]$ where $i\in[1, m]$ and $j\in[1, m-i]$. And D_2 can be represented as:

$$\begin{aligned} D_2 &= \sum_{i=1}^{m} \sum_{j=m-i+1}^{m} a_ib_j(r^{i+j}+r^{-(i+j)}) \\ &= \sum_{i=1}^{m} \sum_{j=m-i+1}^{m} a_ib_j(r^{2m+1-(i+j)}+r^{-(2m+1-(i+j))}) \end{aligned} \tag{6}$$

In Equation (6), the coefficients of β_k in D_2 are the sum of a_ib_j for which $k = 2m+1-|i+j|\in[1, m]$ where $i\in[1, m]$ and $j\in[m-i+1, 2m]$.

Suppose $k(i)=\begin{cases} i \mod 2m+1 & (0\le i \mod 2m+1 \le m) \\ 2m+1-i \mod 2m+1 & (\mathit{others}) \end{cases}$. According to above deduce, it is easy to prove that $\beta_j A=\beta_j\sum_{i=1}^{m} a_i\beta_i=\sum_{i=1}^{m} a_i(\beta_{k(i+j)}+\beta_{k(i-j)})=\sum_{i=1}^{m}(a_{k(i+j)}+a_{k(i-j)})\beta_i$ [19]. Thus, product C also can be represented as:

$$\begin{aligned} C &= A\sum_{j=1}^{m} b_j\beta_j=\sum_{j=1}^{m} b_j(\beta_jA) \\ &= \sum_{j=1}^{m} b_j \sum_{i=1}^{m} a_i(\beta_{k(i+j)}+\beta_{k(i-j)}) \\ &= \sum_{j=1}^{m} b_j \sum_{i=1}^{m} (a_{k(i+j)}+a_{k(i-j)})\beta_i \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} b_j(a_{k(i+j)}+a_{k(i-j)})\beta_i \end{aligned} \tag{7}$$

The single bit c_i can be written as:

$$c_i=\sum_{j=1}^{m} b_j(a_{k(i+j)}+a_{k(i-j)})=\sum_{j=1}^{m} a_j(b_{k(i+j)}+b_{k(i-j)}) \tag{8}$$

From above analysis, II-ONB modular multiplication algorithm based on basis conversion is brought forward, as shown in Algorithm 2. This is a serial algorithm, composed by outer and inner loops. Operations of AND (&), XOR ($\oplus$) and cyclic shift (>>>) can be directly mapped to hardware implementation.

Algorithm 2. II-ONB modular multiplication algorithm based on basis conversion

```
Input: $A = (a'_1, a'_2 \cdots, a'_m), B = (b'_1, b'_2, \cdots, b'_m)$ in normal basis $\underline{N}$
Output: Product $C = (c'_1, c'_2, \cdots, c'_m)$ in normal basis $\underline{N}$
1. Basis conversion : $A = (a_1, a_2 \cdots, a_m), B = (b_1, b_2, \cdots, b_m)$ in basis $\underline{N'}$
2. $C = 0, b_0 = 0$
3. $D[1:2m+1] = \{b_0, b_1, b_2, \cdots, b_m, b_m, \cdots, b_2, b_1\}$
4. for $i = 1$ to $m$ do
5.     for $j = 1$ to $m$ do
6.         $\mathrm{c}_i = \mathrm{c}_i \oplus (a_j \mathbin{\&} (D[j] \oplus D[2m+1-j]))$
       end for
7.     $D >>> 1$ (1 bit cyclic right shift of $D$)
   end for
8. Basis conversion : $C = (c'_1, c'_2, \cdots, c'_m)$ in normal basis $N$
```

In Algorithm 2, input data $A = (a'_1, a'_2 \cdots, a'_m)$ and $B = (b'_1, b'_2, \cdots, b'_m)$ are in normal basis N̲. Firstly, convert operands *A* and *B* from normal basis N̲ to basis N̲'. Then, calculate 1 bit c_i after *m* inner loops and product *C* after *m* outer loops. Finally, convert product *C* from basis N̲' back to normal basis N̲ after the operation is completed. From analysis we can learn that the computational complexity of Algorithm 2 is $O(m^2)$, which can be further improved adopting parallel computation. Therefore, this paper proposes an improved II-ONB modular multiplication algorithm based on basis conversion, as shown in Algorithm 3.

Algorithm 3. An improved II-ONB modular multiplication algorithm based on basis conversion

```
Input: $A = (a'_1, a'_2 \cdots, a'_m), B = (b'_1, b'_2, \cdots, b'_m)$ in normal basis $\underline{N}$
Output: Product $C = (c'_1, c'_2, \cdots, c'_m)$ in normal basis $\underline{N}$
1. Basis conversion : $A = (a_1, a_2 \cdots, a_m)$ , $B = (b_1, b_2, \cdots, b_m)$ in basis $\underline{N'}$
2. $C = 0, b_0 = 0$
3. $D[1:2m+1] = \{b_0, b_1, b_2, \cdots, b_m, b_m, \cdots, b_2, b_1\}$
4. for $i = 1$ to $m$ do
5.     $C = C \oplus a_i \mathbin{\&} (D[1:m] \oplus D[2m:m+1])$
6.     $D >>> 1$ (1 bit cyclic right shift of $D$)
   end for
7. Basis conversion : $C = (c'_1, c'_2, \cdots, c'_m)$ in normal basis $\underline{N}$
```

In Algorithm 3, there exists only one layer loop that can generate product *C* in basis N̲' after *m* loops. So, the computational complexity reduces to $O(m)$. In public-key cryptography, it usually performs continuous modular multiplication operations, so basis conversion is only needed before the first modular multiplication operation and after the last modular multiplication operation. Consequently, the implementing time of basis conversion can be ignored.

4 Design of Reconfigurable II-ONB Modular Multiplier

4.1 Reconfigurable Basis Conversion Circuit

According to above analysis, based on B. Sunar's basis conversion theory, the basis conversion circuit is designed adopting reconfiguration method in this paper. The implementation of basis conversion circuit is closely related to the finite field length *m*. Basis conversion of fixed length can be accomplished by simple connections, but for different lengths, it requires very complex computing circuit. In ECC algorithms, basis conversion is only demanded before and after continuous modular multiplication operations respectively and the implementing time of basis conversion can be ignored, hence it can be realized by SW/HW method. From basis conversion theory we can see the corresponding relationship between *j* and *i* is fixed when *m* is determinate. So in this paper, we calculate corresponding position information in advance by software, then write them into configuration registers. In this way, when performing basis conversion by hardware, choosing values through configuration registers is enough.

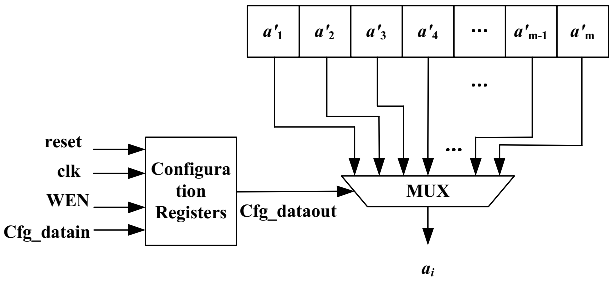

Fig. 1. Reconfigurable basis conversion circuit

The reconfigurable basis conversion circuit is shown in Fig. 1. It requires *m* clock cycles to complete once basis conversion. If hardware resources are adequate, it can use multiple MUXs in parallel to increase basis conversion implementing efficiency. In the best situation, it can complete basis conversion in only one clock cycle.

4.2 Reconfigurable Modular Multiplier Design

In order to support modular multiplication in II-ONB with different lengths, according to Algorithm 3, this paper designs an efficient reconfigurable II-ONB modular multiplier based on basis conversion, as shown in Fig. 2. In the right part, there are two 385-bit registers R1 and R2, both of which are used to store *B* values and b_0 is constant "0". In every one clock cycle, these two registers carry out 1-bit joint shift operation: for R1, $b_i=b_{i+1}$; for R2, $b_i=b_{i-1} \oplus (b_j \,\&\, \text{Datapath\_ctl}[i-1])$. Register C is used to store product C. Datapath_ctl[0:383] is the control signal of the data path, whose value is also associated with the finite field length $\text{Datapath\_ctl} = \{\underbrace{00\cdots0}_{384-\text{length}},1,\underbrace{0\cdots00}_{\text{length-1}}\}$. In the left part, there are two shift registers, which are used to store A and B separately. Their control signals RegA_ctl[7:0] and RegB_ctl[7:0] are given by the control unit. In all the clock cycles, RegA_ctl=RegB_ctl=1.

When signal start is valid, the multiplier begins initialization. The controlled registers R1 and R2 jointly shift 1 bit. At the same time, in order to make the MSB of operation data and registers align right, two barrel shift registers shift (384-length) bits. After length clock cycles, the multiplier generates product C in basis $\underline{N}'$. The maximum length of the multiplier is set 384. When operands are less than 384 bits, all data in registers align right and high bits are filled up zero. According to different length ranges, registers in our design also can be extended to support modular multiplication operation of larger length.

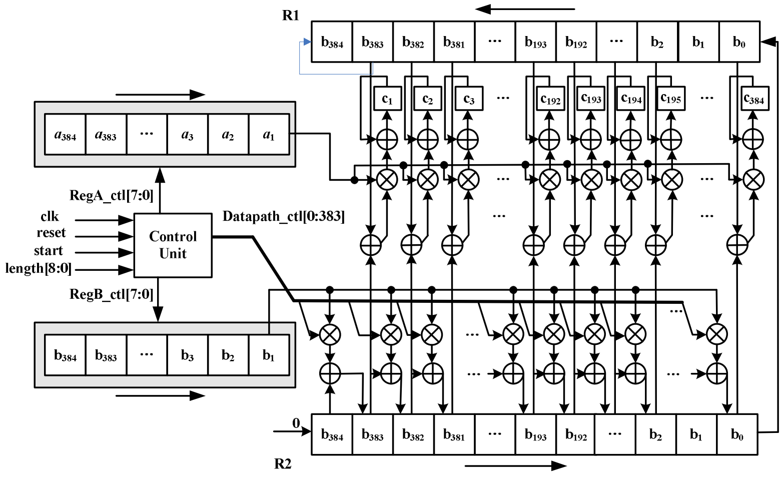

Fig. 2. Reconfigurable II-ONB modular multiplier structure

5 Implementation and Performance Analysis

5.1 Simulation and Synthesis

In order to validate our design, the proposed reconfigurable II-ONB modular multiplier has been modeled in Verilog HDL and simulated functionally with ModelSim SE 6.1f. The implementation has verified our design's function correctly [21]. Fig. 3 depicts simulation results of B-191.

The operation data are:

A= 7a12de5c_d5e55e5a_d587de51_a51c551a_de1b2151_b11a21de
B= 64545d85_5da54d5e_c4b545a5_d45e4456_7aadcccd_dbeebdad

The modular multiplication product is:

C=18db1d7c_d6274cb7_d760e71a_35ae72e5_6de25e4a_73e6fac9

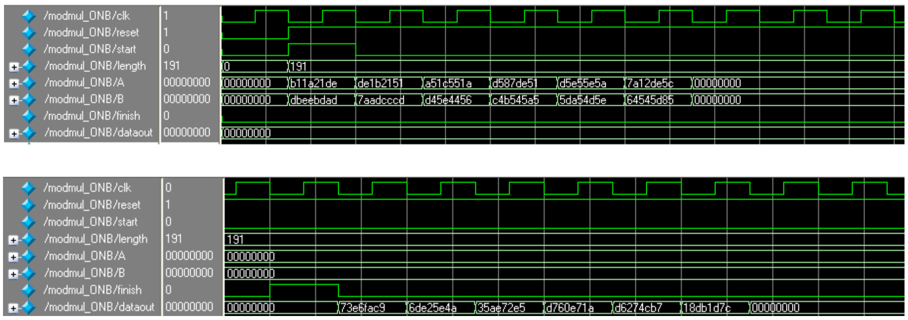
/modmul_ONB/clk 1
/modmul_ONB/reset 1
/modmul_ONB/start 0
/modmul_ONB/length 191 0 191
/modmul_ONB/A 00000000 00000000 b11a21de de1b2151 a51c551a d587de51 d5e55e5a 7a12de5c 00000000
/modmul_ONB/B 00000000 00000000 dbeebdad 7aadcccd d45e4456 c4b545a5 5da54d5e 64545d85 00000000
/modmul_ONB/finish 0
/modmul_ONB/dataout 00000000 00000000
/modmul_ONB/clk 0
/modmul_ONB/reset 1
/modmul_ONB/start 0
/modmul_ONB/length 191 191
/modmul_ONB/A 00000000 00000000
/modmul_ONB/B 00000000 00000000
/modmul_ONB/finish 0
/modmul_ONB/dataout 00000000 00000000 73e6fac9 6de25e4a 35ae72e5 d760e71a d6274cb7 18db1d7c 00000000

Fig. 3. Simulation of B-191 modular multiplication operation

Additional efforts have also been devoted to as ASIC implementation. In order to evaluate performance more accurately, making use of Synopsys's Design Complier for Solaris, logical synthesis has been accomplished under 0.18μm CMOS technology. Table 1 depicts the result report under the constraint of 4.0ns. Our design only occupancies 55k gates in area, while the clock frequency can reach 320MHz.

Table 1. Synthesis results under 0.18μm CMOS technology

Constraint (ns)	Area(μm^2)		Equivalent Gates (kgates)	Delay (ns)
	Combinational Logic	Registers		
4.0	398 868	156 892	55	3.1

5.2 Analysis and Comparison

Area-Timing complexity comparison of different II-ONB modular multipliers are shown in Table 2. The delays of a two-input AND gate and an n-input XOR gate have been approximated by T_A and $\lceil \log_2 n \rceil T_X$ separately. In [19] and [23], multipliers were both designed on word-level, where m denotes the finite field length, k denotes the number of parallel modules and w denotes the word length. Because the hardware structure of our design is fixed, the area and delay of circuit are finalized. The space complexity of this work is 768#AND+1152#XOR and the computational complexity is $2T_A$+$3T_X$. In order to compare our design with others, we have chosen the practical finite field size of m=233 that is a recommended NIST(National Institute of Standards and Technology) binary field degree with k=8 and w=32 which are practical for VLSI implementation. The complexity comparison of different II-ONB modular multipliers in F_2233 with k=8 and w=32 are shown in Table 3.

Table 2. Area-Timing complexity comparison of different II-ONB modular multipliers

Designs	# AND	# XOR	Multiplication Delay
Namin[19]	$2km$	$(4k\text{-}1)m$	$wT_A+(w+\lceil \log_2 2k \rceil)T_X$
Massey[23]	$k(2m\text{-}1)$	$k(2m\text{-}2)$	$w(T_A+(1+\lceil \log_2 m \rceil)T_X)$
This work	768	1152	$2T_A+3T_X$

Table 3. Complexity comparison of different II-ONB modular multipliers in F_2233 with k=8 and w=32

Designs	# of AND	# of XOR	Multiplication Delay
Namin[19]	3728	7223	$32T_A+36T_X$
Massey[23]	3720	3712	$32T_A+288T_X$
This work	768	1152	$64T_A+96T_X$

It can be seen from Table 3 that, due to the word-level structure, compared to our design, in spite of owning shorter multiplication delays, Namin's and Massey-Omura's multipliers occupied much more hardware resources. In addition, both of these two multipliers couldn't support modular multiplication operation with scalable lengths, and it required to re-design the hardware circuit when parameters changed. However, our proposed reconfigure multiplier is one efficient solution for modular multiplications with variable parameters. When parameters changed, only reconfiguring the structural parameters is enough, and the hardware structure doesn't need to change. So, among above designs, our reconfigurable multiplier is the most flexible in structure.

6 Conclusions

In this paper, some researches on reconfiguration design of II-ONB modular multiplier over binary field in ECC have been done. According to B. Sunar's basis conversion theory, operation data in normal basis have been converted to a new defined basis. On this condition, an improved II-ONB modular multiplication algorithm has been proposed, and an efficient reconfigurable modular multiplier supporting different lengths has been implemented. Finally, this work has been simulated and synthesized. Besides, performance analysis has also been accomplished. The experimental results prove that our design has higher flexibility and smaller area, which is the most suitable to resource-constrained ECC applications.

Acknowledgments. This work is partially supported by National Natural Science Foundation of China grant 70890084/G021102 and 61003274, Strategy Pilot Project of Chinese Academy of Sciences sub-project XDA06010702, and National High

Technology Research and Development Program of China (863 Program, No. 2013AA01A214 and 2012AA013104).

References

1. Kammler, D., Zhang, D., Schwabe, P., Scharwaechter, H., Langenberg, M., Auras, D., Ascheid, G., Mathar, R.: Designing an ASIP for Cryptographic Pairings over Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 254–271. Springer, Heidelberg (2009)
2. Gura, N., Shantz, S.C., Eberle, H., Gupta, S., Gupta, V., Finchelstein, D., Goupy, E., Stebila, D.: An End-to-End Systems Approach to Elliptic Curve Cryptography. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 349–365. Springer, Heidelberg (2003)
3. Chen, H.: Research on Elliptic Curve Cryptography Algorithm and Chip Implementation Method, pp. 56-60. Zhengjia University, Hangzhou (2008)
4. Certicom White Papers, The Elliptic Curve Cryptosystem (1997-1998), `http://www.certicom.com`
5. Okada, S., Torii, N., Itoh, K., Takenaka, M.: Implementation of Elliptic Curve Cryptographic Coprocessor over GF(2m) on an FPGA. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 25–40. Springer, Heidelberg (2000)
6. Potgieter, M.J., van Dyk, B.J.: Two Hardware Implementations of the Group operations Necessary for Implementing an Elliptic Curve Cryptosystem over a Characteristic Two Finite Field. In: IEEE Africon 2002, pp. 187–192 (2002)
7. Kitsos, P., Theodoridis, G., Koufopavlou, O.: An Efficient Reconfigurable Multiplier Architecture for Galois Field GF(2m). Microelectronic Journal 34, 975–980 (2003)
8. Amanor, D.N.: Efficient Hardware Architecture for Modular Multiplication. Master. Thesis, The University of Applied Science Offenburg, Germany (2005)
9. Kaihara, M.E., Takagi, N.: Bipartite Modular Multiplication Method. IEEE Transactions on Computers 57(2), 157–164 (2008)
10. Chu, J., Benaissa, M.: Polynomial Residue Number System GF(2m) Multiplier Using Trinomials. In: 17th European Signal Processing Conference (EUSIPCO 2009), pp. 958–962 (2009)
11. Knežević, M., Vercauteren, F., Verbauwhede, I.: Speeding Up Bipartite Modular Multiplication. In: Hasan, M.A., Helleseth, T. (eds.) WAIFI 2010. LNCS, vol. 6087, pp. 166–179. Springer, Heidelberg (2010)
12. Estrin, G., et al.: Parallel Processing in a Restructurable Computer System. IEEE Trans. on Electronic Computers, 747–755 (December 1963)
13. Sigh, H., Lee, M.H., Lu, G., et al.: An Integerated Reconfigurable System for Data-Parallel and Computation-Intensive Applications. IEEE Transcations on Computer 49(5), 465–481 (2000)
14. Bouma, H.: Design and Implementation of an FPGA. University of Twente, Twente (2001)
15. Masoleh, A.R., Hasan, M.A.: Efficient Multiplication beyond Optimal Normal Bases. IEEE Trans. Computers 52(4), 428–439 (2003)
16. Koc, C.K., Sunar, B.: Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields. IEEE Trans. Computers 47(3), 353–356 (1998)
17. Liao, Q.: On Multiplication Tables of Optimal Normal Bases over Finite Fields. Acta Mathematica Sinica 45(5), 947–954 (2005)

18. Sunar, B., Koc, C.K.: An Efficient Optimal Normal Basis Type II Multiplier. IEEE Transactions on Computers 50(1), 83–87 (2001)
19. Namin, A.H., Wu, H., Ahmadi, M.: A High Speed Word Level Finite Field Multiplier Using Reordered Normal Basis. In: IEEE International Symposium on Circuits and Systems, pp. 3278–3281 (2008)
20. Al-Somani, T.F., Amin, A.: High performance Elliptic Curve Point Operations with Pipelined GF(2m) Field Multiplier. Journal of Communication and Computer 6(10), 62–69 (2009)
21. IEEE STD 1363-2000. IEEE Standard Specifications for Public-Key Cryptography (2000)
22. Fang, B., Fan, H., Dai, Y.: An Optimal Normal Basis Type II Multiplier over GF(2n) for FPGAs. Chinese Journal of Electronics 30(12A), 2045–2048 (2002)
23. Massey, J.L., Omura, J.K.: Computational Method and Apparatus for Finite Arithmetic. US: Patent No.4587627 (1986)