World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:5, 2014

Scalable Systolic Multiplier over Binary Extension Fields Based on Two-Level Karatsuba Decomposition

Chiou-Yng Lee, Wen-Yo Lee, Chieh-Tsai Wu, Cheng-Chen Yang

Abstract—Shifted polynomial basis (SPB) is a variation of polynomial basis representation. SPB has potential for efficient bit level and digi -level implementations of multiplication over binary extension fields with subquadratic space complexity. For efficient implementation of pairing computation with large finite fields, this paper presents a new SPB multiplication algorithm based on Karatsuba schemes, and used that to derive a novel scalable multiplier architecture. Analytical results show that the proposed multiplier provides a trade-off between space and time complexities. Our proposed multiplier is modular, regular, and suitable for very large scale integration (VLSI) implementations. It involves less area complexity compared to the multipliers based on traditional decomposition methods. It is therefore, more suitable for efficient hardware implementation of pairing based cryptography and elliptic curve cryptography (ECC) in constraint driven applications.

Keywords—Digit-serial systolic multiplier, elliptic curve cryptography (ECC), Karatsuba algorithm (KA), shifted polynomial basis (SPB), pairing computation.

I. Introduction

Several applications, such as Diffie-Hellman key exchange [1], Digital Signature [2], elliptic curve cryptography (ECC) [3], [4], and pairing-based cryptography [5] use finite field multiplication. The shifted polynomial basis (SPB) [6] provides several advantages for the implementation of finite field multiplications [7]. For pairing-based cryptographic applications, the Weil and Tate pairings based on elliptic curve arithmetic require extensive computations involving operands in large finite fields. For example, 128-bit symmetric-key security could be achieved by computing Tate pairing [8] on a supersingular elliptic curve defined over composite field $GF(2^{4\times 1223})$. Therefore, it is important to explore efficient designs for hardware implementation of pairing computation which is a great challenge, particularly, for the implementation in resource-constrained environments.

Several systolic architectures for multiplication over $GF(2^m)$ are proposed, which could be categorized into bit-parallel and bit-serial architectures. Bit-parallel systolic multipliers perform fast computations, and they are suitable for high-throughout implementations [9], [10]. Nonetheless, such architectures require space-complexity of $O(m^2)$ and typically involve latency of $O(m)$. Bit-serial systolic array multipliers require only $O(m)$ space-complexity [11], [12], but require longer computation time which is not preferable for high-speed applications. To achieve the trade-off between the time and space complexities, digit-serial systolic multipliers with digit-in-digit-out and digit-in-parallel-out structures have been proposed [13], [14]. Moreover, scalable multipliers have also been proposed to achieve a trade-off between time and space complexities. Scalable multipliers [15] are based on fixed $d \times d$ Hankel matrix-vector product (HMVP) approach, which perform the multiplication through $\lceil \frac{m}{d} \rceil^2$ partial products using classical decomposition method. Such kind of scalable feature is used by reconfigurable hardware to decide on the number of partial products to obtain the full multiplication results. Selection of appropriate HMVP structure can lead to less area-delay product compared to classical digit-serial systolic multipliers.

In this paper, we have used SPB for the representation of operands in the proposed algorithm for scalable multiplication over $GF(2^m)$. The proposed multiplication algorithm utilizes two-level KA algorithm, where the outer-level KA performs multi-term decomposition of input polynomials; and the inner-level KA builds parallel systolic multiplier using the products of decomposed input words. The proposed parallel systolic multiplier is realized through $\frac{n^2+n}{2}$ partial products of d-bit words, where $n = \lceil \frac{m}{d} \rceil$. In this paper, we propose a reconfigurable approach to generate necessary operands of those partial products to be used in the proposed scalable systolic SPB multiplier over GF(2^m). Through detail analytical results we have shown that the proposed approach results in a novel scalable systolic array multiplier for fields of large order, which provides a trade-off between space and time complexities.

C.Y. Lee and W.Y. Lee are with Computer Information and Network Engineering, Lunghwa University of Science and Technology, Taoyuan 33306, Taiwan (e-mail: pp010@mail.lhu.edu.tw, TristanWYLee@mail.lhu.edu.tw). C.T. Wu is with Chang Gung Memorial Hospital, Taoyuan, Taiwan. C.C Yang is with Idustrail Technology Research Institute, Hsinchu, Taiwan.

II. Mathematical Background

In this Section, we briefly review the classical SPB multiplication over $GF(2^m)$ and the Karatsuba algorithm.

A. SPB Multiplication over $GF(2^m)$

The ordered set $N = \{1, x, x^2, \cdots, x^{m-1}\}$ is called the polynomial basis of binary extension field $GF(2^m)$, where x is the root of the irreducible polynomial $F(x)$ of degree m. The field element A in $GF(2^m)$ can be represented as $A = a_0 + a_1 x + \cdots + a_{m-1}x^{m-1}$ where $a_i \in \{0, 1\}$ for all i. Let A, B, and C be three elements in $GF(2^m)$, where $C = AB \bmod F(x)$. Generally, the computation of the product C is a two-step operation: (1) the grade-school multiplication

$D = AB$ of at most degree $2m-2$, and (2) the polynomial reduction to compute $C = D \bmod F(x)$.

To represent the elements over $GF(2^m)$, Fan and Dai [6] have defined the SPB of $GF(2^m)$ to derive efficient bit-parallel multiplier for all trinomials as follows:

Definition 1. *Let v be an integer, and let the set $N = \{1, x, x^2, \cdots, x^{m-1}\}$ be the polynomial basis of $GF(2^m)$. Then the ordered set $Nx^{-v} = \{x^{-v}, x^{1-v}, x^{2-v}, \cdots, x^{m-1-v}\}$ is called the shifted polynomial basis of $GF(2^m)$ with respect to N.*

Using the SPB, an element $A = \sum_{i=0}^{m-1} a_i x^i \in GF(2^m)$ can be represented as $\overline{A} = x^{-v}A = \sum_{i=0}^{m-1} a_i x^{i-v}$. It is interesting that there is no hardware cost for the inter-conversion of elements A and $\overline{A}$. For any two elements $\overline{A} = x^{-v}A$ and $\overline{B} = x^{-v}B$ in the SPB representation of $GF(2^m)$, the SPB product $\overline{C} = x^{-v}C$ of $\overline{A}$ and $\overline{B}$ can directly be obtained as

$$C = x^{-v}AB \ mod \ F(x) \tag{1}$$

The two-step computation of SPB multiplication of (1) is described as follows:

Step-1:Grade-school multiplication step

$$T = AB = t_0 + t_1 x + \cdots + t_{2m-2}x^{2m-2} \tag{2}$$

where

$$t_i = \begin{cases} \sum_{k=0}^{i} a_k b_{i-k}, & 0 \le i \le m-1 \\ \sum_{k=i+1-m}^{m-1} a_k b_{m-1-k}, & m \le i \le 2m-2 \end{cases}.$$

Step-2:Polynomial reduction step

$$C = Tx^{-v} \ \bmod \ F(x) \tag{3}$$

According to (3), the complexity of SPB multiplier depends on the chosen value of v. The complexity of SPB multipliers for the field generated by certain types of irreducible polynomials, such as trinomials and pentanomials, is discussed in [6].

B. Multi-term Karatsuba Algorithm

Karatsuba algorithm [16] provides divide-and-conquer technique to multiply long polynomials. It uses three subproducts of half-length operands to replace the original grade-school multiplication. For example, let $A = A_0 + x^{\frac{m}{2}}A_1$ and $B = B_0 + x^{\frac{m}{2}}B_1$ be two polynomials of degree m, where A_0, A_1, B_0, and B_1 are four polynomials of degree $\frac{m}{2}$. The product of A and B can be represented as

$$AB = A_0B_0 + [(A_0 + A_1)(B_0 + B_1). \\ +A_0B_0 + A_1B_1]x^{\frac{m}{2}} + A_1B_1x^m. \tag{4}$$

Based on Karatsuba algorithm, multiplication can be performed in three stages as follows.

1) Evaluation Point (EP) Generation Stage : The polynomial $A = (A_0, A_1)$ is split into the evaluation point vector $EP(A) = (A_0, A_0 + A_1, A_1)$. Similarly, the polynomial $B = (B_0, B_1)$ is also split into $EP(B) = (B_0, B_0 + B_1, B_1)$.
2) Point-Wise Multiplication (PWM) Stage: PWM stage performs point-wise multiplication of $EP(A)$ and $EP(B)$. The PWM is performed after EP generation to produce three products: $D_0 = A_0B_0$, $D_1 = (A_0 + A_1)(B_0 + B_1)$, and $D_2 = A_1B_1$. Thus, we can define that

$$D = PWM(EP(A), EP(B)) = (D_0, D_1, D_2). \tag{5}$$

3) Reconstruction (R) Stage: In this step, the result of the PWM stage is used to construct the desired multiplication result, given by

$$C = (C_0, C_1, C_2) \\ = \mathrm{R}(D) = (D_0, D_0 + D_1 + D_2, D_2). \tag{6}$$

By applying this strategy recursively, each polynomial is transformed into three polynomials with their degrees reduced to about half of its previous polynomial. The decomposition algorithm could be terminated after the polynomials degenerate into single-bit coefficients. The multiplication based on recursive KA scheme is shown in the functional block architecture of Fig. 1. The complexity of KA multiplier is discussed in [17].

For reducing the number of sub-multiplications in the PWM stage, let us define the following identities

$$D_i = A_iB_i, \tag{7}$$

$$D_{ij} = (A_i + A_j)(B_i + B_j). \tag{8}$$

We can compute the product of a pair of three-term polynomials corresponding to 3-term KA scheme. Let A and B be represented by $A = A_0 + A_1x^{m/3} + A_2x^{2m/3}$ and $B = B_0 + B_1x^{m/3} + B_2x^{2m/3}$, respectively, where A_i and B_i are $(\frac{m}{3})$-bit polynomials. The product of A and B can be rewritten

$$C = AB = C_0 + C_1x^{m/3} + C_2x^{2m/3} + C_3x^m + C_4x^{4m/3}, \tag{9}$$

where

$$\begin{aligned} D_0 &= A_0B_0,\ D_1 = A_1B_1,\ D_2 = A_2B_2, \\ D_{01} &= (A_0 + A_1)(B_0 + B_1), \\ D_{12} &= (A_2 + A_1)(B_2 + B_1), \\ D_{02} &= (A_0 + A_2)(B_0 + B_2), \\ C_0 &= D_0,\ C_1 = D_{01} + D_0 + D_1, \\ C_2 &= D_{02} + D_0 + D_1 + D_2, \\ C_3 &= D_{12} + D_1 + D_2,\ C_4 = D_2. \end{aligned}$$

In the above example, three-term KA requires 6 multiplications of decomposed operands to generate the partial products and 13 additions of decomposed operands and of partial products. Generalizing the above multi-term KA decomposition, we can obtain the following properties.

Lemma 1. *Suppose A and B are two m-bit polynomials. Based on n-term KA scheme with one-step approach, both polynomials A and B are split into $(\frac{m}{n})$-bit subword polynomials. In this case, to compute $C = AB$, we require $\frac{n^2+n}{2}$ partial products.*

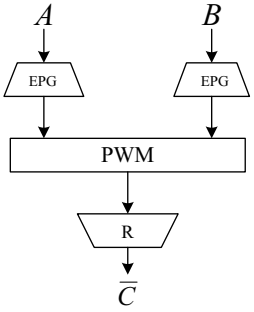

Fig. 1. The functional block of the KA architecture.

In the structure of Fig. 1, it is shown that three stages are performed sequentially and based on that we can have Lemma 2 in the following.

Lemma 2. *Assume that $C = C_1 + C_2$, where $C_1 = A_1B_1$ and $C_2 = A_2B_2$. In the three stage computation of Fig.1, C_1 and C_2 can be represented by $C_1 = R(PWM(EP(A_1), EP(B_1)))$ and $C_2 = R(PWM(EP(A_2), EP(B_2)))$, respectively. $C = C_1 + C_2$ is then directly obtained by the following recombination structure:*

$$C = R(PWM(EP(A_1), EP(B_1)) + PWM(EP(A_2), EP(B_2)). \quad (10)$$

III. Proposed Scalable Systolic SPB Multiplier

In this Section, we utilize the multi-way KA scheme to derive a novel scalable multiplier over $GF(2^m)$, where the field element is represented by the SPB.

A. Parallel Systolic Array for Computing Partial Product Multiplication

Assume that $\overline{A} = \sum_{i=0}^{n-1} \bar{a}_i x^i$ and $\overline{B} = \sum_{i=0}^{n-1} \bar{b}_i x^i$ are two subword polynomials, and $\overline{C}$ be their product prior to the reduction operation, such that

$$\overline{C} = \overline{AB} = \bar{b}_0\overline{A} + \bar{b}_1 x\overline{A} + \cdots + \bar{b}_{n-1}x^{n-1}\overline{A}. \quad (11)$$

For digit-size d , operand $\overline{B}$ could be decomposed into p number of d-bit sub-words with $p = \lceil \frac{n}{d} \rceil$, such that

$$\overline{B} = \sum_{i=0}^{p-1} \overline{B}_i x^{id}, \quad (12)$$

where

$$\overline{B}_i = \sum_{j=0}^{d-1} \bar{b}_{id+j} x^j.$$

Thus, the product $\overline{C}$ can be represented as

$$\overline{C} = \sum_{i=0}^{p-1} \overline{AB}_i x^{id}. \quad (13)$$

Next, according to multi-term KA algorithm (Sec. II-B), we can use d-term KA to derive the partial product $\overline{C}$ in (9). According to the structure of Fig. 1, the product of two d-bit polynomials can be constructed in three stages, such as evaluation stage, point-wise multiplication stage, and reconstruction stage. We have the following complexities for each of these stages.

Lemma 3. *Assume that A and B are two d-bit polynomials, and the product of A and B is based on 1-step d-term KA scheme. The components in Fig. 1 then have the following time and space complexities:*

- Each EP circuit requires $\frac{d^2-d}{2}$ XOR gates and involves one XOR gate delay.
- The PWM circuit requires $\frac{d^2+d}{2}$ XOR gates and involves one AND gate delay.
- The R circuit requires $(d^2 - d)$ XOR gates and involves $\lceil \log_2(1.5d) \rceil$ XOR gate delays.

For the selected digit-size d, the element $\overline{A}$ is represented by $\overline{A} = \overline{A}_0 + \overline{A}_1 x^d + \cdots + \overline{A}_{p-1}x^{d(p-1)}$, where $\overline{A}_i = \sum_{j=0}^{d-1} \overline{a}_{id+j} x^j$ are the sub-words. Let us denote the evaluation point of $\overline{A}$ given by $P_{\overline{A}} = EP(\overline{A})$, which could be rewritten as

$$P_{\overline{A}} = P_{\overline{A}_0} + P_{\overline{A}_1} x^d + \cdots + P_{\overline{A}_{p-1}} x^{d(p-1)} \quad (14)$$

Utilizing the recombination property (stated in Lemma 2), the partial product C_i in (9) can be obtained as

$$\overline{C} = \mathrm{R}(\sum_{i=0}^{p-1} \mathrm{PWM}(P_{\overline{A}}, P_{\overline{B}_i}) x^{id}), \quad (15)$$

where

$$\mathrm{PWM}(P_{\overline{A}}, P_{\overline{B}_i}) = \sum_{j=0}^{p-1} PWM(P_{\overline{A}_j}, P_{\overline{B}_i}) x^{dj}$$

Fig. 2 shows the signal-flow graph (SFG) for the computation of (15) based on one-step KA decomposition. In Fig.2, the EP-A module is comprised of p number of EP circuits, the PWM-A module is comprised of p PWM circuits to perform $\mathrm{PWM}(P_{\overline{A}}, P_{\overline{B}_i})$, and the final reconstruction (FR) requires $(2p - 1)$ R circuits. We can use Lemma 3 to evaluate the time complexities of EP, PWM and R circuits as T_X, T_A and $\log_2(1.5d)T_X$ gate delays, respectively. To achieve the minimum critical path, we assume that k and l are two positive integers which satisfy $k = \lceil \frac{p}{l} \rceil$, where $l = 2^{\lceil \log_2(1.5d) \rceil - 2}$. For example, if the multiplier selects 5-bit digit-size to implement the partial product, then we have $l = 2$. We can use cut-set retiming with $l = 2$ (as shown in Fig.2) to reduce the delay between PWM-EP-adder and FR. Thus, the product $\overline{C}$ is selected by l PWM-A modules given by the following formula

$$\overline{C} = R(\sum_{i=0}^{k-1} \overline{C}_i x^{ild}) \quad (16)$$

where

$$\overline{C}_i = \mathrm{PWM}(P_{\overline{A}}, P_{\widetilde{B}_i}) = \sum_{j=0}^{l-1} \mathrm{PWM}(P_{\overline{A}}, P_{\overline{B}_{il+j}}) x^{jd}$$

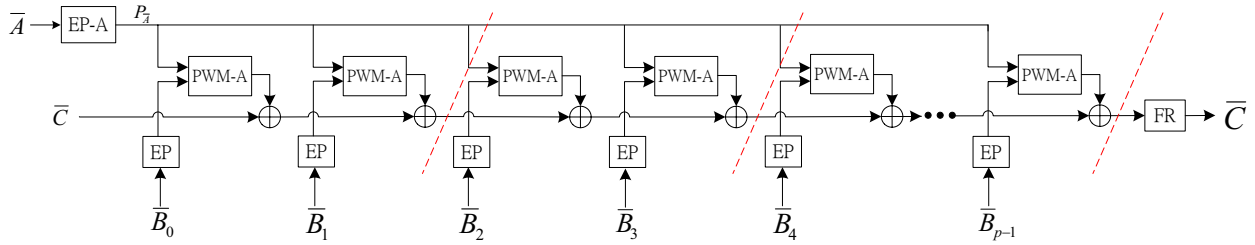

Fig. 2. Signal-flow graph (SFG) for computing partial products

$$= \sum_{j=0}^{l-1}\sum_{t=0}^{p-1} \text{PWM}(P_{\overline{A}_t}, P_{\overline{B}_{il+j}})x^{(t+j)d} \quad (17)$$

$$\widetilde{B}_i = \overline{B}_{il} + \overline{B}_{il+1}x^d + \cdots + \overline{B}_{il+l-1}x^{d(l-1)}. \quad (18)$$

In order to reduce the latency, let us consider a pair of integers t and w which satisfy $w = \lceil \frac{k}{t} \rceil$. The product $\overline{C}$ in (16) can then be expressed as

$$\overline{C} = R(\sum_{i=0}^{w-1} \overline{\overline{C}}_i x^{ildt}) \quad (19)$$

where

$$\overline{\overline{C}}_i = \sum_{i=0}^{t-1} \overline{C}_i x^{ild}. \quad (20)$$

Fig.3 shows the proposed 2-D parallel systolic array for computing the partial products based on (19). It consists of one EP-A, w parallel systolic arrays (PSA), one pipelined adder-tree (PAT), and one FR. Each PSA is composed of t PEs to implement (20). Fig.4 shows the design of PE, which is composed of one PWM multiplier core, one adder, and one EP-B. Each PWM multiplier core performs the computation of (17). It consists of pl PWMs. The EP-B is based on (18) to compute the evaluation point $P_{\widetilde{B}_i}$. It consists of l EPs. Let us define $S_\otimes = \frac{d(d+1)}{2}$. The EP-B module in Fig.4 produces $lS_\otimes$ output bits, and the PWM multiplier core produces $(p+l-1)S_\otimes$-bits of results.

B. Final Polynomial Reduction circuit

After the degree alignment operation, we get the result D which is a $(2m-1)$-bit polynomial. The most significant $m-1$ terms of D are recursively reduced through polynomials of degree less than m using the irreducible polynomial $F(x)$ to obtain $C = x^{-v}D \bmod F(x)$. In [18], it is shown that $C = x^{-v}D \bmod F(x)$ can be represented by

$$C = [I_{m\times m}|Q][d_0, d_1, \cdots, d_{2m-2}]^T.$$

where Q is the reduction matrix associated with the irreducible polynomial $F(x)$. For any general reduction polynomial $F(x)$, the final polynomial reduction (FPR) module requires $\text{H}(Q)$ XOR gates and $\log_2(\theta+1)T_X$ critical path, where $\text{H}(Q)$ is the Hamming weight of the reduction matrix Q, and θ is the the maximum Hamming weight of the column vectors in matrix Q. For the NIST recommended irreducible polynomials for elliptic curve cryptosystems [2], Table I lists the complexity of FPR module.

TABLE I. COMPLEXITY OF FINAL POLYNOMIAL REDUCTION (FPR) FOR TRINOMIALS AND PENTANOMIALS

polynomials	#XOR(H(Q))	time delay
$x^m + x^n + 1$	$2m-2$	$2T_X$
$x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$ for $1 < k_1 < k_2 < k_3 < m/2$	$4m-4$	$4T_X$
$x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$ for $1 < k_1 < k_2 < k_3 < m/2$ and $k_3 - k_2 = k_1$	$3m + k_1 - 3$	$4T_X$

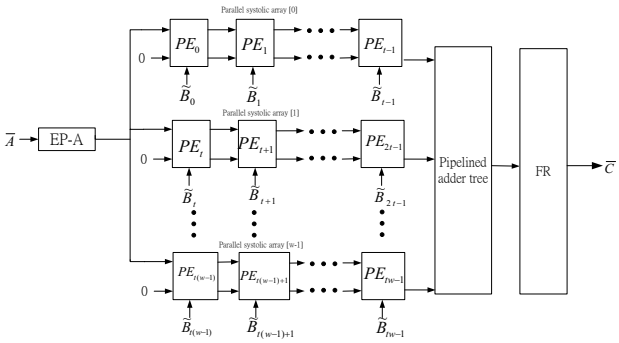

Fig. 3. The proposed parallel systolic array architecture for computing partial products

C. Proposed Scalable Architecture

Let the field element $A = \overline{a}_0 + \overline{a}_1 x + \cdots + \overline{a}_{m-1}x^{m-1}$ in $GF(2^m)$ be represented by $A = A_0 + A_1 x^n + A_2 x^{2n}$, where $n = \lceil \frac{m}{3} \rceil$ and $A_i = a_{i,0} + a_{i,1}x + \cdots + a_{i,n-1}x^{n-1}$ for $0 \le i \le 2$ and $a_{i,j} = \overline{a}_{ni+j}$, $0 \le j \le n-1$. Let the field $GF(2^m)$ be constructed from an irreducible polynomial $F(x)$ of degree m. For $A, B \in GF(2^m)$, the SPB product $C = RAB \bmod F(x)$ can be represented as

$$C = R[A_0B_0 + (A_0B_0 + A_1B_1 + A_{01}B_{01})x^n$$
$$+(A_0B_0 + A_1B_1 + A_2B_2 + A_{02}B_{02})x^{2n}$$
$$+(A_1B_1 + A_2B_2 + A_{12}B_{12})x^{3n} + A_2B_2x^{4n}] \mod F(x)$$
$$= R[A_0B_0(1 + x^n + x^{2n}) + A_1B_1(x^n + x^{2n} + x^{3n})$$

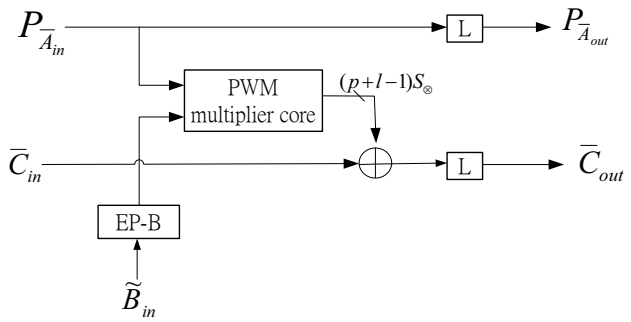

Fig. 4. Functional diagram of processing element (PE)

$$+A_2B_2(x^{2n}+x^{3n}+x^{4n})+A_3B_3x^n$$
$$+A_4B_4x^{2n}+A_5B_5x^{3n}] \mod F(x) \quad (21)$$

where

$$A_3 = A_0 + A_1, A_4 = A_0 + A_2, A_5 = A_1 + A_2,$$
$$B_3 = B_0 + B_1, B_4 = B_0 + B_2, B_5 = B_1 + B_2.$$

From (21), we can find that the product C includes six partial products: $C_0 = A_0B_0$, $C_1 = A_1B_1$, $C_2 = A_2B_2$, $C_3 = A_3B_3$, $C_4 = A_4B_4$, and $C_5 = A_5B_5$. We need to generate the decomposed operands A_i and B_i from the operands A and B, for i =0,1, .., 5, according to the following relations.

$$A_i = (s_{i,00}A_0 + s_{i,01}A_1 + s_{i,02}A_2)$$
$$+(s_{i,10}A_0 + s_{i,11}A_1 + s_{i,12}A_2). \quad (22)$$
$$B_i = (s_{i,00}B_0 + s_{i,01}B_1 + s_{i,02}B_2)$$
$$+(s_{i,10}B_0 + s_{i,11}B_1 + s_{i,12}B_2). \quad (23)$$

where $S_{i,0} = (s_{i,00}, s_{i,01}, s_{i,02})$ and $S_{i,1} = (s_{i,10}, s_{i,11}, s_{i,12})$ are used to determine the decomposed operands A_i and B_i. Each partial product $C_i = A_iB_i$ is required to be multiplied with a sparse polynomial P_i for $i = 0, 1, \cdots, 5$, given by $P_0 = 1+x^n+x^{2n}$, $P_1 = x^n+x^{2n}+x^{3n}$, $P_2 = x^{2n}+x^{3n}+x^{4n}$, $P_3 = x^n$, $P_4 = x^{2n}$ and $P_5 = x^{3n}$. We can define $P_i = s_{i,20}+s_{i,21}x^n+s_{i,22}x^{2n}+s_{i,23}x^{3n}+s_{i,24}x^{4n}$ to generate the sparse polynomials. Table II lists three control vectors for per cycles to determine the partial products. The partial products are computed sequentially in the order C_0, C_1, C_2, C_3, C_4, C_5, and multiplied with the corresponding sparse polynomials and summed together to obtain the intermediate product to be reduced thereafter.

According to Table II, Algorithm 1 can be used for scalable SPB multiplication based on three-way KA scheme. In this algorithm, steps 4 and 5 are performed to decompose the subword polynomials $\overline{A}_i$ and $\overline{B}_i$, respectively. Step 6 is based on Sec.III-A to carry out subword multiplication. Step 7 performs the degree-alignment operation. Finally, Step 9 does the final polynomial reduction. Fig. 5 shows the proposed SPB/GPB multiplier with scalable hardware implementation according to Algorithm 1, where the different operands are generated in different configurations realized by

Algorithm 1 Proposed scalable SPB multiplication algorithm based on three-way KA

Inputs: $A = A_0+A_1x^n+A_2x^{2n}$ and $B = B_0+B_1x^n+B_2x^{2n}$ are two element in GF(2^m), where $n = \lceil \frac{m}{3} \rceil$. R is a nonzero polynomial.
Output: $C = RAB \mod F(x)$.
1. $D = 0$.
2. Generate three control vector tables S_0, S_1, S_2.
3. for $i = 0$ to 5
4. $\overline{A}_i = (s_{i,00}A_0+s_{i,01}A_1+s_{i,02}A_2)+(s_{i,10}A_0+s_{i,11}A_1+s_{i,12}A_2)$.
5. $\overline{B}_i = (s_{i,00}B_0+s_{i,01}B_1+s_{i,02}B_2)+(s_{i,10}B_0+s_{i,11}B_1+s_{i,12}B_2)$.
6. $\overline{C}_i = \overline{B}_i\overline{A}_i$.
7. $D = D + (s_{i,20} + s_{i,21}x^n + s_{i,22}x^{2n} + s_{i,23}x^{3n} + s_{i,24}x^{4n})\overline{C}_i$.
8. endfor
9. $C = DR \mod F(x)$.

the control vectors S_0, S_1 and S_2. The proposed architecture consists of one control unit, three registers (A, B, D registers), two decomposed operand generation circuits, one subword polynomial multiplier (Fig.3), one degree-alignment circuit, and one FPR.

According to (21), the KA for three-way decomposition involves 6 partial products C_i for $0 \le i \le 5$, and all partial products involve the multiplication of different combinations of input subwords. Three control vectors S_0, S_1 and S_2 are stored in the circular shift-register in the control unit. During each iteration of partial product computations, the decomposed operand generation circuit (using a pair of control vectors S_0 and S_1) produces the corresponding input polynomials for the subword polynomial multiplier for computing the partial products. The degree-alignment circuit selects one of the six different terms for product reconstruction using the control vector S_2. For example, in Table II, we select the three control vectors $S_{3,0} = (100)$, $S_{3,1} = (010)$, $S_{3,2} = (01000)$ to perform the computation of $(A_0 + A_1)(B_0 + B_1)x^n$ in (21), as shown in the path diagram by the red lines in Fig. 5. Based on this approach, we use three control vectors $S_{2,0} = (001)$, $S_{2,1} = (000)$ and $S_{2,2} = (00111)$ to calculate $A_2B_2(x^{2n} + x^{3n} + x^{4n})$, and so on.

IV. Time and Space Complexities

In Fig.5 we have used two-level KA decomposition to implement the proposed scalable SPB multiplier. The proposed architecture is composed of two decomposed operand generation circuits, one parallel systolic multiplier, one degree alignment circuit, one final reduction circuit, and three registers. The decomposed operand generation circuit (Fig.5) is based on the outer-level KA scheme to produce the low-order polynomials. The parallel systolic multiplier is used to compute product of low-order decomposed polynomials obtained from the inner-level KA scheme. The degree-alignment circuit is used to perform the reconstruction function to obtain product word according to the outer-level KA

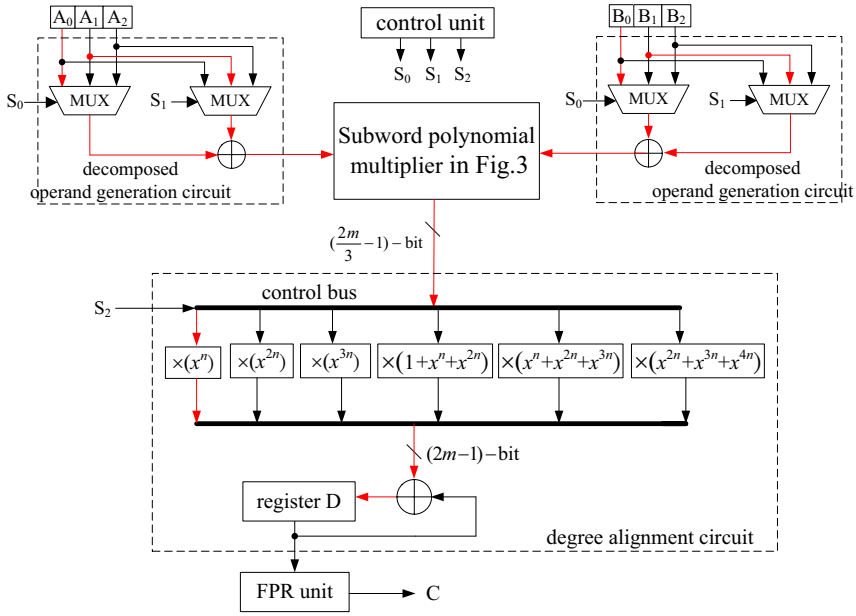

Fig. 5. Proposed scalable multiplier using 3-way KA.

TABLE II. THREE CONTROL TABLES

(a) S_0 CONTROL TABLE

i	s_{00}	s_{01}	s_{02}
0	1	0	0
1	0	1	0
2	0	0	1
3	1	0	0
4	1	0	0
5	0	1	0

(b) S_1 CONTROL TABLE

i	s_{10}	s_{11}	s_{12}
0	0	0	0
1	0	0	0
2	0	0	0
3	0	1	0
4	0	0	1
5	0	0	1

(c) S_2 CONTROL TABLE

i	s_{20}	s_{21}	s_{22}	s_{23}	s_{24}
0	1	1	1	0	0
1	0	1	1	1	0
2	0	0	1	1	1
3	0	1	0	0	0
4	0	0	1	0	0
5	0	0	0	1	0

scheme. Two-level KA is used in one-step d-term KA to develop our proposed scalable multiplier. Tables III and IV show space and time complexities, respectively. In Table IV, it is shown that, if we choose the d-term KA scheme, the proposed architecture needs $\frac{d^2+d}{2}$ partial products, while the corresponding scalar multipliers [19], [20] require d^2 partial products. As the number of terms in KAs increases, the partial products used in Fig. 3 could be reduced and hence the hardware complexity of implementation could be reduced. Therefore, the proposed multiplier can efficiently make a trade-off between space and time complexities.

For pairing computation with the 128-bit security level, the field $GF(2^{1223})$ constructed by the trinomial $x^{1223}+x^{255}+1$ is used. We, therefore, choose this field to estimate critical-path,

TABLE IV. COMPARISON OF TIME COMPLEXITIES OF MULTIPLIERS

Multipliers	Latency	critical path
[22]	$2\sqrt{\frac{m}{d}}$	$T_A + (\log_2(d+1))T_X$
[20], [19]	$d^2 + 2n - 2$	$T_A + T_X$
[23]	$2\left\lceil \frac{m}{d} \right\rceil$	$T_A + (\log_2 d)T_X + T_M$
Fig.5	$\frac{d^2+d}{2} + t + 4$	$T_A + (log_2(l) + 2)T_X$

Note:(1)$wt = \left\lceil \frac{m}{d^2 l} \right\rceil$, and d is the selected digit-size.
(2) T_A, T_X, and T_M denote the propagation delays of a 2-input AND gate, a 2-input XOR gate, and a 2×1 MUX gate, respectively.

area complexity, and area-delay product of digit-serial/scalable systolic multipliers. To make a fair comparison, we have used the NanGate's Library Creator and the 45-nm FreePDK Base Kit from North Carolina State University (NCSU) [21] to synthesize the proposed and the corresponding existing digit-serial multipliers and obtained time and area complexities. As shown in Table V, for digit-size $d = 10$, the area×delay product (ADP) of our proposed architecture is significantly lower than those of the existing multipliers. Amongst all the existing digit-serial/scalable systolic multipliers, Lee's multiplier [22] has the minimum time-complexity. But as shown in Table V, the proposed multiplier involves less area-complexity and area-delay product (ADP) compared with those of [22]. When the number of terms in outer-level decomposition is increases, our proposed scalable multiplier involves the lowest area-complexity amongst the digit-wise systolic multipliers [20], [13], [22]. Therefore, our proposed architecture can achieve a tradeoff between space and time complexities.

TABLE III. COMPARISON OF SPACE COMPLEXITIES OF MULTIPLIERS

Multipliers	Basis	structure	#AND	#XOR	#MUX	#Latch
[22]	PB	digit-serial	$m\sqrt{m}$	$\sqrt{md}(2+m)+d$		$\sqrt{\frac{m}{d}}(2m+d-1)+2m$
[20]	Montgomery	scalable	n^2	n^2+3n-1	$n+2$	$2n^2+5dn-2d+n$
[19]	DB	scalable	n^2	n^2+2n	$dn+n$	$2n^2+2dn+2d$
[23]	PB	digit-serial	md	$md+2d$	$2m$	$4m+3d+1$
Fig.5	SPB	scalable	$2m+\frac{n^2}{d^2}S_{\otimes}-1$	$\approx 2m+2n+d+\frac{2n}{d}S_{\oplus}$ $+S_{\otimes}(\frac{4n}{d}+\frac{n^2}{d^2})+H(Q)$	$4m$	$2m+3n+(\frac{n^2}{d^2}+(\frac{n}{d})^{1.5})S_{\otimes}$

Note:(1) d is the selected digit-size, and is also the number of splitting terms. (2) $n=\lceil\frac{m}{d}\rceil$, $S_{\oplus}=\frac{d^2-d}{2}$, and $S_{\otimes}=\frac{d^2+d}{2}$. (3) $H(Q)$ is the Hamming weight of the reduction matrix Q for an irreducible polynomial.

TABLE V. COMPARISON OF VARIOUS DIGIT-SERIAL/SCALABLE MULTIPLIERS OVER $GF(2^{1223})$ IN TERMS OF LATENCY, TOTAL CRITICAL DELAY $T_{TCD}(ns)$, AREA (μm^2), AREA×DELAY PRODUCT (ADP)$(\mu m^2)ns$ FOR DIGIT-SIZE $d=10$

multipliers	Latency	T_{TCD}	area	ADP
[22]	24	6.24	406,855	2,538,778
[20]	344	48.16	206,166	9,928,952
[19]	344	48.16	191,196	9,208,012
[23]	246	73.8	59,379	4,382,242
Fig.5	62	16.12	100,713	1,623,496

V. Conclusions

In this paper, we propose a new scalable systolic SPB multiplier over $GF(2^m)$. We have derived a SPB multiplication algorithm and its architecture to realize the proposed scalable systolic multiplier. To explore the area-time trade-off for large field arithmetic architectures, we have used two-level Karatsuba schemes to implement the scalable systolic multiplier over $GF(2^{1223})$. As the number of terms in the Karatsuba method increases, it involves significantly less area and ADP. The analytical results provide a valuable reference for implementing pairing algorithm and elliptic curve digital signature algorithm (ECDSA) in resource-constrained embedded systems and smart phones. Moreover, our proposed multiplier has regularity and modularity which make it suitable of VLSI realization.

ACKNOWLEDGEMENT

The work is supported by the Taiwan National Science Council (NSC) under Grant NSC 1022221-E-262-018.

References

[1] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.
[2] "Digital Signature Standard," *National Institute of Standards and Technology*, 186-2, Jan. 2000.
[3] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
[4] A. Menezes, I. Blake, S. Gao, R. Mullin, S. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*. Kluwer Academic Publisher, 1993.
[5] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.
[6] H. Fan and M. Hasan, "Fast Bit Parallel Shifted Polynomial Basis Multipliers in $GF(2^n)$," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 53, no. 12, pp. 2606–2615, 2006.
[7] N. Mentens, S. B. Ors, B. Preneel, and J. Vandewalle, "An FPGA Implementation of a Montgomery multiplier over $GF(2^m)$," in *Proc. of the 7th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2004, pp. 121–128.
[8] D. Kammler, D. Zhang, P. Schwabe, H. Scharwaechter, M. Langenberg, D. Auras, G. Ascheid, and R. Mathar, "Designing an asip for cryptographic pairings over barreto-naehrig curves," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, ser. Lecture Notes in Computer Science, vol. 5747. Springer, Heidelberg, 2009, pp. 254–271.
[9] P. K. Meher, "Systolic and Super-Systolic Multipliers for Finite Field $GF(2^m)$ Based on Irreducible Trinomials," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 55, no. 4, pp. 1031–1040, May 2008.
[10] 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 Trans. Computers*, vol. 50, no. 5, pp. 385–393, May 2001.
[11] G. N. Selimis, A. P. Fournaris, H. E. Michail, and O. Koufopavlou, "Improved Throughput Bit-Serial Multiplier for GF(2^m) Fields," *Integration, the VLSI journal*, vol. 42, pp. 217–226, 2009.
[12] S. Fenn, M. Gossel, M. Benaissa, and D. Taylor, "On-Line Error Detection for Bit-Serial Multipliers in $GF(2^m)$," *Journal of Electronic Testing: Theory and Applications*, vol. 13, no. 1, pp. 29–40, 1998.
[13] S. Talapatra, H. Rahaman, and J. Mathew, "Low complexity Digit Serial Systolic Montgomery Multipliers for Special Class of $GF(2^m)$," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 5, pp. 487–852, May 2010.
[14] M. Morales-Sandoval, C. Feregrino-Uribe, and P. Kitsos, "Bit-serial and digit-serial $GF(2^m)$ Montgomery multipliers using linear feedback shift registers," *IET Computers and Digital Techniques*, vol. 5, no. 2, pp. 86–94, 2011.
[15] C.-Y. Lee and C. Chiou, "Scalable Gaussian Normal Basis Multipliers over $GF(2^m)$ Using Hankel Matrix-Vector Representation," *Journal of Signal Processing Systems*, vol. 69, no. 2, pp. 197–211, 2012.
[16] A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata," *ISoviet Physics-Doklady (English translation)*, vol. 7, no. 7, pp. 595–596, 1963.
[17] C. Paar, "A new architecture for a parallel finite field multiplier with low complexity based on composite fields," *IEEE Trans. Computers*, vol. 45, no. 7, pp. 856–861, Jul. 1996.
[18] A. Reyhani-Masoleh and M. Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over $GF(2^m)$," *IEEE Trans. Computers*, vol. 53, no. 8, pp. 945–959, 2004.
[19] L. H. Chen, P. L. Chang, C.-Y. Lee, and Y. K. Yang, "Scalable and Systolic Dual Basis Multiplier over $GF(2^m)$," *Intl Journal of Innovative Computing, Information and Control*, vol. 7, no. 3, pp. 1193–1208, Mar. 2011.
[20] C.-Y. Lee, C. W. Chiou, J. M. Lin, and C. C. Chang, "Scalable and Systolic Montgomery Multiplier over $GF(2^m)$ Generated by Trinomials," *IET Circuits, Devices & Systems*, vol. 1, no. 6, pp. 477–484, 2007.
[21] "Nangate standard cell library," *http://www.si2.org/openeda.si2.org/projects/nangatelib/*.
[22] C.-Y. Lee, "Super Digit-Serial Systolic Multiplier Over $GF(2^m)$," in *Sixth International Conference on Genetic and Evolutionary Computing*, 2012.
[23] S. Talapatra, H. Rahaman, and S. K. Saha, "Unified Digit Serial Systolic Montgomery Multiplication Architecture for Special Classes of Polynomials over GF(2^m)," in *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 2010.