Efficient Subquadratic Space Complexity Digit-Serial Multipliers over $GF(2^m)$ based on Bivariate Polynomial Basis Representation

Chiou-Yng Lee

Computer Information & Network Engineering
Lunghwa University of Science & Technology
Taoyuan, Taiwan 333
e-mail: pp010@gm.lhu.edu.tw

Jiafeng Xie

Electrical & Computer Engineering
Villanova University
Villanova, PA 19010, USA
e-mail: jiafeng.xie@villanova.edu (corresponding author)

Abstract— Digit-serial finite field multipliers over $GF(2^m)$ with subquadratic space complexity are critical components to many applications such as elliptic curve cryptography. In this paper, we propose a pair of novel digit-serial multipliers based on bivariate polynomial basis (BPB). Firstly, we have proposed a novel digit-serial BPB multiplication algorithm based on a new decomposition strategy. Secondly, the proposed algorithm is properly mapped into a pair of pipelined and non-pipelined digit-serial multipliers. Lastly, through the detailed complexity analysis and comparison, the proposed designs are found to have less area-time complexities than the competing ones.

I. Introduction

Finite field multiplier over $GF(2^m)$ is a crucial component in many cryptographic applications such as elliptic curve cryptography [1], [2], [3], [4] and discrete-log-based cryptography [5]. Since polynomial basis multiplication provides smaller computational complexity than the other bases [6], quite a number of reports have been made on efficient implementation of finite field multipliers on this basis [7].

Recently, digit-serial multipliers have gained substantial attentions from the research community: (i) the digit-serial structures have better time-complexity than the bit-serial ones; (ii) compared with the bit-parallel multipliers, the digit-serial designs have smaller area-complexity. Thus, many useful approaches have been proposed to reduce the complexities of the digit-serial structures [8], [9].

Among all these released reports, Karatsuba algorithm (KA) [10] and Toeplitz matrix-vector product (TMVP) are recognized as the two most effective methods to reduce the computational complexity of the finite field multipliers. Novel (a, b)-way KA decomposition is proposed in [8] to obtain efficient realization. Another report has presented a k-partition scheme combined with TMVP approach to obtain an area-delay efficient digit-serial multiplier [11]. In [12], it is shown that the (a, b)-way KA block recombination (KABR) can provide a better tradeoff between time and space complexities. Recently, a more efficient block TMVP approach is introduced in [13]. Two newly reports use KA strategy to obtain ultra low-complexity finite field multipliers [14], [15]. All these efforts represent the major advance in the field.

Following this direction, in this paper, we propose to present novel digit-serial multipliers based on a new decomposition strategy. We observe that the newly released bivariate polynomial basis (BPB) multiplication [16] has attractive characteristic that can be utilized for more efficient implementation, i.e., the last two steps of the BPB multiplication (split into three steps) involve regular multiplication process and can be decomposed further. Based on this observation, in this paper, we propose three stages of interdependent efforts: (i) a novel digit-serial BPB multiplication algorithm (based on a new KABR (NKABR) approach) is proposed first to obtain subquadratic space complexity; (ii) a pair of new digit-serial multipliers are then efficiently mapped from the proposed algorithm; and (iii) a detailed comparison (complexity analysis) is carried out to confirm the efficiency of the proposed designs.

The rest of the paper is organized as follows. Section II briefly gives the preliminaries. In Section III, we propose the novel digit-serial multipliers mapped from the proposed algorithm. In Section IV, we analyze the performance of the proposed structures along with the existing designs. Finally, we conclude the paper in Section V.

II. Preliminaries

A. KA: Karatsuba algorithm

Let n be a power of 2 and the polynomial U be $U = \sum_{i=0}^{n-1} u_i x^i$ over $GF(2)$. Then, let $U = U_0 + U_1 x^{n/2}$, where $U_j = \sum_{i=0}^{\frac{n}{2}-1} u_{i+\frac{n}{2}j} x^i$ for $j = 0$ and 1. Likewise, $V = \sum_{i=0}^{n-1} v_i x^i$ over $GF(2)$ can also be $V = V_0 + V_1 x^{n/2}$. Thus, the product $Z = UV$ is

$$\begin{aligned} Z =&(U_0 + U_1 x^{n/2})(V_0 + V_1 x^{n/2}) \\ =&U_0 V_0 (1 + x^{\frac{n}{2}}) + U_{01} V_{01} x^{\frac{n}{2}} + U_1 V_1 (x^{\frac{n}{2}} + x^n), \end{aligned} \tag{1}$$

where $U_{01} = U_0 + U_1,\ V_{01} = V_0 + V_1$. We can then define four components (two evaluate points (EPs), one point-wise product (PW), and one reconstruction (R)) as

$$\begin{cases} \text{EP1} = \{U_0, U_{01}, U_1\} \\ \text{EP2} = \{V_0, V_{01}, V_1\} \\ \text{PW}(P) = \{P_0, P_1, P_2\} \\ C = \text{R}(\text{PW}(P)) = \{P_0, P_0 + P_1 + P_2, P_2\}, \end{cases} \tag{2}$$

where $P_0 = U_0 V_0, P_1 = U_{01} V_{01}, P_2 = U_1 V_1$.

The three multiplications of (1) can be calculated recursively by further decomposition based on (2). The corresponding complexities of this decomposition are in [8], [9].

B. BPB: bivariate polynomial basis multiplication

Definition 1. Let $y = x^n$, then the polynomial $A = \sum_{i=0}^{m-1} a_i x^i$ can be $A = \sum_{i=0}^{n-1} A_i x^i$, where $A_i = \sum_{j=0}^{k-1} a_{i,j} y^j$, $k = \lceil \frac{m}{n} \rceil$, and m is divisible by n. The set $\{1, y, \cdots, y^{k-1}, x, xy, \cdots, xy^{k-1}, \cdots, x^{n-1}y, \cdots, x^{n-1}y^{k-1}\}$ is called the BPB representation [16], [17].

When m is not divisible by n, we need to pad $(nk-m)$-bit zeros at the least significant bit of A to satisfy the definition of the BPB, i.e., $\overline{A} = x^{kn-m}A$. To use BPB, we must modify the reduction polynomial $F(x)$ as follows.

Definition 2. Let $F(x) = x^m + \sum_{i=0}^{q} f_i x^i$ ($q < m/2$). With BPB representation, we have $\overline{F}(x) = x^{kn-m}F(x) = y^k + G(x)$, where $G(x) = x^{kn-m}(\sum_{i=0}^{q} f_i x^i) = \sum_{i=0}^{n-1} g_i(y) x^i$ ($y = x^n$ and $k = \lceil \frac{m}{n} \rceil$).

Example 1. We use the field $GF(2^{14})$ to illustrate the modified polynomial $F(x)$: Assume that $F(x) = x^{14} + x^5 + 1$ is used to construct the field $GF(2^{14})$. Let us define $y = x^4$, we can obtain that $\overline{F}(x) = x^2 F(x) = x^{16} + x^7 + x^2 = y^4 + x^3 y + x^2$.

Let A and B be represented by the BPB, the product $C = AB = (\sum_{i=0}^{n-1} A_i x^i)(\sum_{j=0}^{n-1} B_j x^j)$ over $GF(2^m)$ satisfies $y = x^n$ and $y^k = \sum_{i=0}^{n-1} g_i(y) x^i$. The BPB multiplication can thus be expressed by Algorithm 1 as

Algorithm 1 The BPB multiplication algorithm [16], [17]

```
Input: $A$ and $B \in GF(2^m)$, $n$ is a positive integer.
Output: $C = AB \bmod F(x)$
1. Step-I (multiplication):
$T = AB$
2. Step-II (reduction):
$D = \sum_{i=0}^{n-1} D_i(y) x^i = T \bmod (y + x^n)$
3. Step-III (reduction):
$C = \sum_{i=0}^{n-1} C_i x^i = \mathrm{SR}(D)$ (reduction in $y$)
```

Step-I. $T = AB = \sum_{i=0}^{2n-2} T_i x^i$, where $T_i = \sum_{j+k=i} A_j B_k$ (the degree of each T_i is at most $(2k-2)$).

Step-II. Based on the bivariate polynomial $y + x^n$, we have $x^{n+i} = x^i y$, for $0 \le i \le n-2$. Then, we have ($D = \sum_{i=0}^{n-1} D_i x^i = T \bmod (y + x^n)$)

$$\begin{aligned} D &= \sum_{i=0}^{n-1} T_i x^i + x^n \sum_{i=0}^{n-2} T_{n+i} x^i \bmod (y + x^n) \\ &= \sum_{i=0}^{n-1} T_i x^i + y \sum_{i=0}^{n-2} T_{n+i} x^i, \end{aligned} \tag{3}$$

which can be represented as

$$\begin{aligned} \begin{bmatrix} D_0 \\ D_1 \\ \vdots \\ D_{n-1} \end{bmatrix} &= \begin{bmatrix} A_0 & yA_{n-1} & \cdots & yA_1 \\ A_1 & A_0 & \cdots & yA_2 \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-1} & A_{n-2} & \cdots & A_0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ \vdots \\ B_{n-1} \end{bmatrix} \\ &= M_A B, \end{aligned} \tag{4}$$

where we can find that the degree of D_i in y for $0 \le i \le n$ is smaller than or equal to $(2k-1)$ since the degrees of A_i and B_j are $(k-1)$, and the degree of yA_i is k. The product D (matrix-vector form) can be decomposed further.

Step-III. In this step, we can use the polynomial $y^k + \sum_{i=0}^{n-1} g_i(y) x^i$ to reduce subword D_i. Assume that we split subword D_i into two parts $D_i^{(0)}$ and $D_i^{(1)}$ with respect to the degree in y of its coefficients

$$D_i = D_i^{(0)} + y^k D_i^{(1)}, \tag{5}$$

where $D_i^{(0)} = \sum_{j=0}^{k-1} d_{i,j}^{(0)} y^j$ and $D_i^{(1)} = \sum_{j=0}^{k-2} d_{i,j}^{(1)} y^j$.

The product D can be re-expressed as

$$D = \overline{D}_0 + y^k \overline{D}_1, \tag{6}$$

where $\overline{D}_0 = D_0^{(0)} + D_1^{(0)} x + \cdots + D_{n-1}^{(0)} x^{n-1}$ and $\overline{D}_1 = D_0^{(1)} + D_1^{(1)} x + \cdots + D_{n-2}^{(1)} x^{n-2}$. Based on the polynomials $y^k + \sum_{i=0}^{n-1} g_i(y) x^i$ and $y = x^n$, we can have

$$\begin{aligned} y^k &= g_0(y) + g_1(y)x + \cdots + g_{n-1}(y)x^{n-1} \\ xy^k &= yg_{n-1}(y) + g_0(y)x + \cdots + g_{n-2}(y)x^{n-1} \\ &\cdots\ \cdots\ \cdots \\ x^{n-1}y^k &= yg_1(y) + yg_2(y)x + \cdots + g_0(y)x^{n-1}. \end{aligned} \tag{7}$$

Then, $y^k \overline{D}_1$ can be

$$\begin{aligned} y^k \overline{D}_1 &= \begin{bmatrix} g_0(y) & yg_{n-1}(y) & \cdots & yg_1(y) \\ g_1(y) & g_0(y) & \cdots & yg_2(y) \\ \vdots & \vdots & \ddots & \vdots \\ g_{n-1}(y) & g_{n-2}(y) & \cdots & g_0(y) \end{bmatrix} \begin{bmatrix} D_0^{(1)} \\ D_1^{(1)} \\ \vdots \\ D_{n-1}^{(1)} \end{bmatrix} \\ &= M_g \overline{D}_1, \end{aligned} \tag{8}$$

which can be extended to have

$$C = \overline{D}_0 + M_g \overline{D}_1, \tag{9}$$

when $\overline{F}(x) = y^k + G(x)$ is a trinomial/pentanomial, we can have $g_i(y) \in \{0, 1, y\}$. Following Example 1, $\overline{F}(x) = x^2 F(x) = y^4 + x^3 y + x^2$, we have $g_0(y) = 0, g_1(y) = 0$, $g_2(y) = 1$, $g_3(y) = y$. For the proposed multiplication algorithm, we use $GF(2^m)$ to extend the field $GF(2^{nk})$.

III. Proposed Digit-Serial Multipliers

We observe that the two reduction steps of the BPB multiplication (Algorithm 1) involve regular multiplication process, as shown by the matrix-vector product forms of (6) and (9). Here, we propose a novel BPB multiplication algorithm to utilize this unique property to obtain efficient subquadratic space complexity BPB multipliers.

A. NKABR: new Karatsuba algorithm block recombination approach

Connected with (1), product Z can be rewritten as

$$Z = \mathrm{R}(W_1) + \mathrm{R}(W_2) = \mathrm{R}(W_1 + W_2), \tag{10}$$

where $W_1 = \mathrm{PW}(\mathrm{EP1}(U_1) \overline{\otimes} \mathrm{EP2}(V_1))$ and $W_2 = \mathrm{PW}(\mathrm{EP1}$ $(U_2) \overline{\otimes} \mathrm{EP2}(V_2))$ (symbol $\overline{\otimes}$ denotes the PW). As the PW function produces S_A^{PW} bits (assume $S(\cdot)$ and $E(\cdot)$ denote the space and delay complexities, respectively; $()_X$ and $()_A$ represent the related logic gates, i.e., XOR & AND; T_A and T_X are the AND and XOR gates' delay, respectively), the component addition (CA) for the addition of W_1 and W_2 involves $S_X^{\mathrm{CA}}(n) = S_A^{\mathrm{PW}}$ (where $S_X^{\mathrm{CA}}(n) < S_X^{\mathrm{R}}(n)$). Thus,

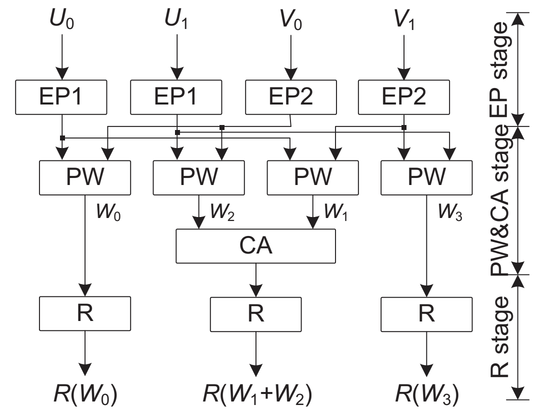

Fig. 1. High-level structure of the proposed NKABR approach.

the space complexity of the NKABR $Z = \mathrm{R}(W_1 + W_2)$ is reduced to $2S_X^{\mathrm{EP1}} + 2S_X^{\mathrm{EP2}} + 2S_A^{\mathrm{PW}} + S_X^{\mathrm{CA}} + S_X^{\mathrm{R}}$, which is smaller than the original KA of $Z = \mathrm{R}(W_1) + \mathrm{R}(W_2)$. The example below gives detailed process of the NKABR method.
Example 2. For 1-iterative NKABR approach, we have

$$\begin{aligned} Z =&(U_0 + U_1x^{n/2})(V_0 + V_1x^{n/2}) \\ =&U_0V_0 + (U_0V_1 + U_1V_0)x^{n/2} + U_1V_1x^n \\ =&\mathrm{R}(W_0) + \mathrm{R}(W_1 + W_2)x^{n/2} + \mathrm{R}(W_3)x^n, \end{aligned} \tag{11}$$

where $W_0 = \mathrm{PW}(\mathrm{EP1}(U_0)\overline{\otimes}\mathrm{EP2}(V_0))$, $W_1 = \mathrm{PW}(\mathrm{EP1}(U_0)\overline{\otimes}\mathrm{EP2}(V_1))$, $W_2 = \mathrm{PW}(\mathrm{EP1}(U_1)\overline{\otimes}\mathrm{EP2}(V_0))$, and $W_3 = \mathrm{PW}(\mathrm{EP1}(U_1)\overline{\otimes}\mathrm{EP2}(V_1))$. The corresponding structure is shown in Fig. 1.

B. Proposed BPB multiplication algorithm

Observing the matrix-vector product of (4), let us define that the i-th column of M_A be the polynomial $A^{(i)}$, where $A^{(0)} = A = \sum_{i=0}^{n-1} A_ix^i$. Based on the definition of matrix M_A, we can find that the computation of $A^{(i)}$ equals $A^{(i)} = A^{(i-1)}x$ if $A^{(i-1)}$ is predetermined. Note that $A^{(i)}$ must use two reduction steps to transform the BPB representation.
First reduction (FR) step: Let us define that $\mathrm{FR}(A) = A \bmod (y + x^n)$. We can use FR to reduce $A^{(i)}$ in x as

$$\begin{aligned} A^{(i)} =&\mathrm{FR}(xA^{(i-1)}) \\ =&yA_{n-1}^{(i-1)} + A_0^{(i-1)}x + \cdots + A_{n-2}^{(i-1)}. \end{aligned} \tag{12}$$

Note that $A^{(i)}$ in (12) is equal to the i-th column of matrix M_A. Based on (12), we can obtain that $A^{(i)} = \mathrm{FR}(xA^{(i-1)}) = \mathrm{FR}(x^iA)$.
Second reduction (SR) step: Based on (12), we must use the polynomial $\overline{F}(x) = y^k + G(x) = y^k + \sum_{i=0}^{n-1} g_ix^i$ to reduce $yA_{n-1}^{(i-1)}$ in y. Let $A_{n-1}^{(i-1)} = a_{0,n-1}^{(i-1)} + a_{1,n-1}^{(i-1)}y + \cdots + a_{n-1,n-1}^{(i-1)}y^{k-1} = \overline{A}_{n-1}^{(i-1)} + a_{n-1,n-1}^{(i-1)}y^{k-1}$, where $\overline{A}_{n-1}^{(i-1)} = a_{0,n-1}^{(i-1)}y + a_{1,n-1}^{(i-1)}y + \cdots + a_{n-2,n-1}^{(i-1)}y^{k-2}$. Based on the irreducible polynomial $\overline{F}(x) = y^k + G(x) = y^k + \sum_{i=0}^{n-1} g_ix^i$, we can have $y^k = \sum_{i=0}^{n-1} g_ix^i$. Therefore, $yA_{n-1}^{(i-1)}$ following the polynomial reduction $\overline{F}(x)$ can be obtained as

$$yA_{n-1}^{(i-1)} \bmod \overline{F}(x) = y\overline{A}_{n-1}^{(i-1)} + a_{n-1,n-1}^{(i-1)}G(x). \tag{13}$$

Based on (13), the polynomial $A^{(i)}$ can be

$$\begin{aligned} A^{(i)} &= (y\overline{A}_{n-1}^{(i-1)} + a_{n-1,n-1}^{(i-1)}g_0(y)) \\ &+ \sum_{j=1}^{n-1}(A_{j-1}^{(i-1)} + a_{n-1,n-1}^{(i-1)}g_j(y))x^j \\ &= \sum_{j=0}^{n-1} A_j^{(i)}x^j. \end{aligned} \tag{14}$$

According to (4), the product D can be rewritten as

$$D = A^{(0)}B_0 + A^{(1)}B_1 + \cdots + A^{(n-1)}B_{n-1}, \tag{15}$$

where the partial product $A^{(j)}B_i = A_0^{(j)}B_i + A_1^{(j)}B_ix + \cdots + A_{n-1}^{(j)}B_ix^{n-1}$ is the core operation of D.
NKABR employing strategy. We then propose a novel NKABR employing strategy to realize (15), i.e., all the partial products $A_g^{(j)}B_i$ $(0 \le g \le n-1)$ are innovatively computed and recombined with the basic NKABR module of order a^i: (i) employ the NKABR method of Fig. 1 to decompose each sub-product $A_g^{(j)}B_i$ into three levels of computation (two EPs, one PW&CA, and one R); (ii) let n NKABR modules work in parallel to obtain the partial product $A^{(j)}B_i$; (iii) use the SR step of (8) to get the final result $C = D \bmod \overline{F}(x)$ after the product D is obtained from serial accumulation of $A^{(j)}B_i$. The above detailed processes are summarized in Algorithm 2.

Algorithm 2 Proposed subquadratic digit-serial multiplication algorithm based on the NKABR strategy

```
Input: A and B are two BPB element in $GF(2^m)$
Output: $C = AB \bmod F(x)$
1. Initial step:
1.1. $\mathrm{EP1} = \mathrm{EP2} = W = D = \mathrm{R} = 0$
1.2. $A = A_0 + A_1x + \cdots + A_{n-1}x^{n-1}$
1.3. $B = B_0 + B_1x + \cdots + B_{n-1}x^{n-1}$
2. Multiplication step:
2.1. for $i = 0$ to $n-1$
2.2. $\mathrm{EP1} = (\mathrm{EP1}(A_0), \mathrm{EP1}(A_1), \cdots, \mathrm{EP1}(A_{n-1}))$
2.3. $\mathrm{EP2} = \mathrm{EP2}(B_i)$
2.4. $W = W + \mathrm{PW\&CA}(\mathrm{EP1}, \mathrm{EP2})$
2.5. $A = \mathrm{SR}(\mathrm{FR}(xA))$
2.6. end for
3. Reconstruction (R) step:
3.1 $D = \mathrm{R}(W)$
4. Second reduction (SR) step:
4.1. $C = \mathrm{SR}(D)$
```

C. Proposed pipelined digit-serial multiplier

Based on Algorithm 2, Fig. 2 shows the proposed digit-serial BPB multiplier, which is executed through four stages of operations (t_0, t_1, t_2, and t_3). As shown in Fig. 2, the core component of the proposed structure is realized by the z-iterative NKABR approach with order $k = a^i$ and $z < i$. Moreover, we have also used the T-type flip-flops (FFs) to replace the accumulation circuits built with XOR gates followed by the D-type FFs, i.e., the T-type FF (loaded with $< W >$) performs $W = W + \mathrm{PW\&CA}(\mathrm{EP1}, \mathrm{EP2})$ of Step 2.4 of Algorithm 2 to reduce the involved complexity.

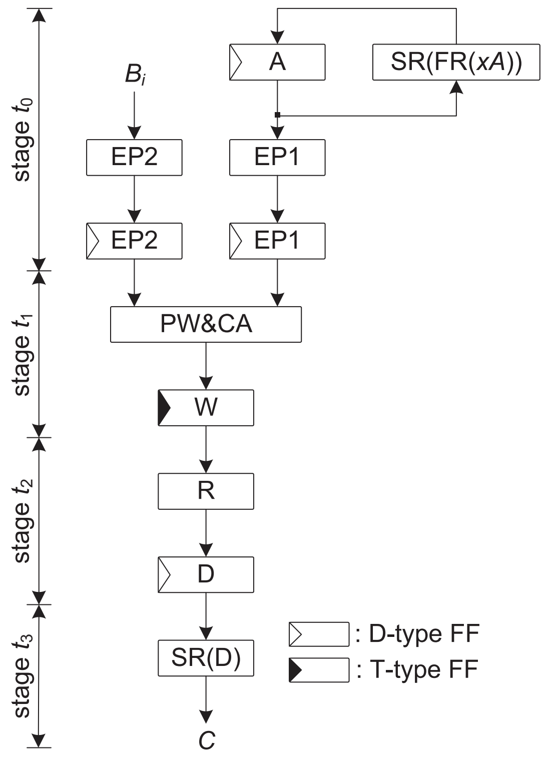

Fig. 2. The proposed pipelined digit-serial BPB multiplier.

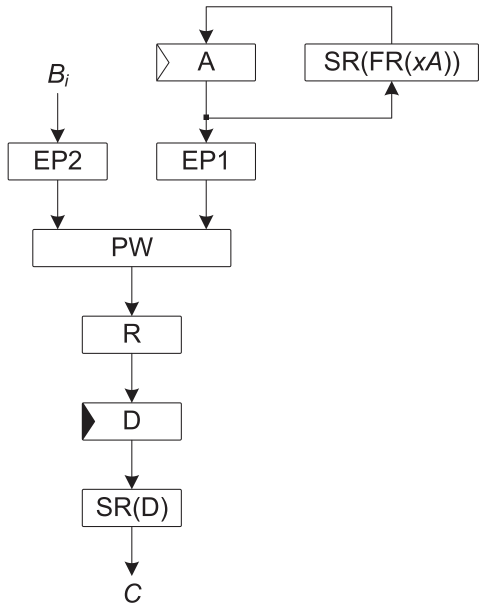

Fig. 3. The proposed non-pipelined digit-serial BPB multiplier.

At the initial step, the register $< A >$ is loaded with the operand A. Then, following the multiplication step of Algorithm 2, the intermediate result is stored in the register $< W >$, which requires $(n+1)$ clock cycles. After the multiplication step is fully executed, there is a need of extra 2 clock cycles to finish the R step (Step 3 of Algorithm 2) and the SR step (Step 4 of Algorithm 2). In total, the proposed digit-serial bivariate multiplication involves $(n+3)$ clock cycles, and the critical-path delay (CPD) is $MAX(\mathrm{CPD}_{t_0}, \mathrm{CPD}_{t_1}, \mathrm{CPD}_{t_2}, \mathrm{CPD}_{t_3})$.

Stage t_0. This stage involves one EP1 component, one EP2 component, one $\mathrm{SR}(\mathrm{FR}(Ax))$ unit, and three registers ($< \mathrm{EP1} >$, $< \mathrm{EP2} >$, and $< A >$). For z-iterative a-way NKABR approach (uses a^z EP2 components to split polynomial B_i and na^z EP1 components to split polynomial A), we can find that EP2 component involves $a^z S_X^{\mathrm{EP1}}(\frac{k}{a^z})$ XOR gates and EP1 component involves $na^z S_X^{\mathrm{EP1}}(\frac{k}{a^z})$ XOR gates (requires a delay of $E_X^{\mathrm{EP1}}(\frac{k}{a^z})T_X$). Two registers $< \mathrm{EP2} >$ and $< \mathrm{EP1} >$ involve $a^z S_A^{\mathrm{PW}}(\frac{k}{a^z})$ and $na^z S_A^{\mathrm{PW}}(\frac{k}{a^z})$ FFs, respectively. The register $< A >$ contains nk FFs, the $\mathrm{SR}(\mathrm{FR}(Ax))$ unit involves 2 XOR gates and T_X delay for trinomial-based multiplication (4 XORs and T_X for pantanomial-based one). The overall complexity is (based on trinomial)

$$\begin{cases} \mathrm{XOR} = (n+1)a^z S_X^{\mathrm{EP1}}(\frac{k}{a^z}) + 2 \\ \mathrm{FF} = (n+1)a^z S_A^{\mathrm{PW}}(\frac{k}{a^z}) + nk \\ \mathrm{CPD}_{t_0} = E_X^{\mathrm{EP1}}(\frac{k}{a^z})T_X. \end{cases} \tag{16}$$

Stage t_1. There are one PW&CA module and one register $< W >$ involved in this stage, where the PW&CA operation involves na^{2z} PW components and $n(a^{2z} - 2a^z + 1)$ CA components and register $< W >$ involves $n(2a^z - 1)S_A^{\mathrm{PW}}(\frac{k}{a^z})$ FFs. Besides, one can replace the AND gates with NAND gates since $y = (a_0a_1) \oplus (a_2a_3)$ is the same as $y = (\overline{a_0a_1}) \oplus (\overline{a_2a_3})$. In total, we have

$$\begin{cases} \mathrm{XOR} = n(2a^z - 1)S_X^{\mathrm{CA}}(\frac{k}{a^z}) \\ \mathrm{NAND} = na^{2z} S_A^{\mathrm{PW}}(\frac{k}{a^z}) \\ \mathrm{FF} = n(2a^z - 1)S_A^{\mathrm{PW}}(\frac{k}{a^z}) \\ \mathrm{CPD}_{t_1} = T_A + zT_X. \end{cases} \tag{17}$$

Stage t_2. This stage contains one R component and one register $< D >$, where the R component involves $n(2a^z - 1)S_X^{\mathrm{R}}(\frac{k}{a^z})$ XORs (a delay of $E_X^{\mathrm{R}}(\frac{k}{a^z})T_X$) and the register $< D >$ has $n(2k-1)$ FFs.

Stage t_3. The content of the register $< D >$ is $\overline{D}_0 + y^k\overline{D}_1$ according to (6). Based on (8), $\mathrm{SR}(D)$ is computed as $C = D \bmod \overline{F}(x) = \overline{D}_0 + M_g\overline{D}_1$, where the complexity of $M_g\overline{D}_1$ depends on the chosen polynomial $\overline{F}(x) = y^k + \sum_{i=0}^{n-1} g_i(y)$. If $\overline{F}(x)$ is an irreducible trinomial, then $\mathrm{SR}(D)$ involves at least $(2nk - 2n)$ XOR gates and a delay of $2T_X$ ($(4nk - 6n)$ XOR gates and a delay of $3T_X$ for pentanomial based structure).

D. Proposed non-pipelined digit-serial multiplier

. Fig. 3 shows the structure of the proposed non-pipelined digit-serial multiplier, where the original register $< D >$ is realized by the T-type FF (the original CA components of Fig. 2 are removed and the addition operations are then realized by the T-type FFs following the computation process of Step 2.4 of Algorithm 2). The corresponding complexities of the non-pipelined digit-serial multiplier are

$$\begin{cases} \mathrm{XOR} = (n+1)a^z S_X^{\mathrm{EP1}}(\frac{k}{a^z}) + n(2a^z - 1)S_X^{\mathrm{CA}}(\frac{k}{a^z}) \\ \qquad + n(2a^z - 1)S_X^{\mathrm{R}}(\frac{k}{a^z}) + Q \\ \mathrm{NAND} = na^{2z} S_A^{\mathrm{PW}}(\frac{k}{a^z}) \\ \mathrm{FF} = n(2k-1) \\ \mathrm{CPD} = T_A + (z + E_X^{\mathrm{EP1}}(\frac{k}{a^z}) + E_X^{\mathrm{R}}(\frac{k}{a^z}))T_X, \end{cases} \tag{18}$$

TABLE I
COMPARISON OF AREA-TIME COMPLEXITIES OF THE PROPOSED STRUCTURES AND THE EXISTING DIGIT-SERIAL MULTIPLIERS

Design	Latency	CPD	XOR	AND	NAND	FF
[9]	$n^{\log_a \frac{a}{2}}+1$	$T_A+3\log_a nT_X$	S_0+Q	$n^{\log_a \frac{3a}{2}}$	-	$2n+d$
[12]	$\frac{1}{\sqrt{2}}m^{\log_4 2}+1$	$T_A+(2+3\log_4(m/2))T_X$	S_1+Q	$1.63m^{\log_4 2}$	-	$2m+m^{\log_4 2}$
[18]	$\lceil\frac{m}{d}\rceil$	$T_A+(\lceil\log_2(d+1)\rceil+\lceil\log_2 T\rceil)T_X$	$d(mT-m-T+2)$	md	-	$2m$
[19]	$\lceil\frac{m}{d}\rceil+1$	$T_A+(1+\lceil\log_2(d+1)\rceil)T_X$	$d(m+k-1)$ $+(d-1)(k+1)$	$dm+d(k-1)$ $+(d-1)(k+1)$	-	$2m+d+k$
[11]	$\lceil\frac{m}{d}\rceil+1$	$T_A+(2+\log_2(a-1)\log_a d)T_X$	S_2+Q	$\lceil\frac{m}{d}\rceil d^{\log_a \frac{a^2+a}{2}}$	-	$2n+d$
Fig. 2	$\lceil\frac{m}{d}\rceil+3$	CPD_1	S_3+Q	-	$\lceil\frac{m}{d}\rceil a^{2z}S_A^{\mathrm{PW}}(\frac{d}{a^z})$	S_4
Fig. 3	$\lceil\frac{m}{d}\rceil+1$	$T_A+(z+E_X^{\mathrm{EP1}}(\frac{k}{a^z})+E_X^{\mathrm{R}}(\frac{k}{a^z}))T_X$	S_3+Q	-	$\lceil\frac{m}{d}\rceil a^{2z}S_A^{\mathrm{PW}}(\frac{d}{a^z})$	$3m-1$

d is the selected digit-size satisfying $d=a^i$ for $i>1$, Q is the complexity of the reduction polynomial $F(x)=x^m+\sum_{i=0}^{k} f_i x^i$.
$S_0=(3+\frac{2a+2}{3a-4}-\frac{2a}{3a-2})n^{\log_a \frac{3a}{2}}-3n-\frac{2a+2}{3a-4}n^{\log_a 2}+\frac{2a}{3a-2}$, where $n\geq m$.
$S_1=4.12m^{\log_4 6}-2.5m-0.77m^{\log_4 2}+0.4$.
$S_2=\lceil\frac{m}{d}\rceil[(5-\frac{2(a-3)}{a(a-1)}-\frac{2}{(a-1)(a+2)})d^{\log_a \frac{a(a+1)}{2}}-(5-\frac{2(a-3)}{a(a-1)})d+\frac{2}{(a-1)(a+2)}]$.
$S_3=(n+1)a^z S_X^{\mathrm{EP1}}(\frac{k}{a^z})+2+n(2a^z-1)S_X^{\mathrm{CA}}(\frac{k}{a^z})+n(2a^z-1)S_X^{\mathrm{R}}(\frac{k}{a^z})+2nk-2n$.
$S_4=(\lceil\frac{m}{d}\rceil+1)a^z S_X^{\mathrm{EP1}}(\frac{d}{a^z})+\lceil\frac{m}{d}\rceil(a^{2z}-2a^z+1)S_X^{\mathrm{CA}}(\frac{d}{a^z})+\lceil\frac{m}{d}\rceil(2a^z-1)S_X^{\mathrm{R}}(\frac{d}{a^z})$.
$S_5=((3\lceil\frac{m}{d}\rceil+1)a^z-\lceil\frac{m}{d}\rceil)S_A^{\mathrm{PW}}(\frac{d}{a^z})+3m-\lceil\frac{m}{d}\rceil$.
$\mathrm{CPD}_1=MAX(E_X^{\mathrm{EP1}}(\frac{k}{a^z})T_X, T_A+z, E_X^{\mathrm{R}}(\frac{k}{a^z})T_X, \mathrm{CPD}_{\mathrm{SR}})$.
$S_X^{\mathrm{EP1}}(\frac{d}{a^z})$, $a^z S_A^{\mathrm{PW}}(\frac{d}{a^z})$, and $S_X^{\mathrm{R}}(\frac{d}{a^z})$ are determined by the specific reduction polynomial size and can be obtained from Table II of [12].
[18] is type-T Gaussian normal basis multiplier (even T).
Due to the complicated complexity expression, the complexities of [13], [14], [15] are not listed here.

where Q is the space complexity of the reduction polynomial.

IV. COMPLEXITY & COMPARISON

A. Complexity

The area-time complexities, in terms of latency cycles, CPD, logic gates count, and registers, of the proposed designs and the existing ones of [9], [12], [18], [19], [11] are shown in Table I. Due to limited space, the complexities of [13], [14], [15] are not listed (but are included in the comparison).

B. Comparison

While it is complicated to demonstrate the actual efficiency of the proposed designs, we have coded the proposed multipliers with VHDL and have estimated their complexities on both application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) platforms following the comparing strategies of the existing designs. We have used the NanGate's Library Creator and the 45-nm FreePDK Base Kit from North Carolina State University (NCSU) [20] and the Intel Straix-V 5SGXMA9N1F45C2 device (Intel Quartus Prime 17.0). We have also chosen the relatively large reduction polynomial $F(x) = x^{409} + x^{87} + 1$ (suggested by the National Institute of Standards and Technology (NIST)) and two different digit-size of $d = 32$ and $d = 16$. To have a fair comparison, we have used the two-parallel (4,2)-way KA decomposition [9] with order 256 for digit-size 16; we have used 28-parallel (4,2)-way KAs of order 64 for digit-size 16, and 8-parallel (4,2)-way KAs of order 256 for digit-size 32 to realize the KABR structure of [12]; while the proposed structure is realized by the 2-way NKABR scheme, where we have chosen the 2-iterative NKABR approach for $d = 16$ and 3-iterative NKABR method for $d = 32$.

Considering the fact that the designs of [9], [12], [13], [18], [19], [11] are mainly reported in the form of area-time complexities while [14], [15] are largely targeted on FPGA devices with extra power consumption information, we thus have faithfully followed their styles and presented the corresponding complexities of various multipliers with respect to different d as shown in Tables II and III, respectively.

One can see that the proposed multipliers obtain the best area-time complexities among all the designs. For example, the proposed design (Fig. 3, $d = 32$) involves at least 26.1% less area than the existing designs and also at least 43.5% smaller ADP than the best competing ones, as seen from Table II. While comparing with those KA-based designs implemented on FPGA device, such as [14], [15], one can see that the proposed designs significantly outperform these newly reported ones.

C. Discussion

The proposed designs have better tradeoff in overall complexities than the existing ones. For most of the cases, the proposed designs also have relatively smaller area complexity brought by the proposed NKABR method. Future work can be extended to novel small-complexity multipliers for post-quantum cryptography.

V. CONCLUSION

A novel digit-serial BPB multiplication algorithm based on the NKABR approach is proposed in this paper. With the help of proper mapping strategies, we have presented a pair of efficient digit-serial multipliers with subquadratic space complexity. The detailed complexity and comparison have shown that the proposed designs significantly outperform the existing ones.

TABLE II
COMPARISON OF AREA-TIME COMPLEXITIES OF VARIOUS DIGIT-SERIAL MULTIPLIERS OVER $GF(2^{409})$ IN TERMS OF LATENCY (CYCLES), TOTAL CRITICAL DELAY $T_{\text{TCD}}(ns) = T_{\text{CPD}} \times$ Latency Cyles, AREA (μm^2), AND AREA-DELAY PRODUCT (ADP)$(\mu m^2 \cdot ns) =$ Area $\times T_{\text{TCD}}$.

Design	Digit-size	Latency	T_{TCD}	Area	ADP
[9]	16	27	15.7	17,983	281,625
[12]	32	14	7.56	28,825	217,923
	16	27	13.5	17,281	233,300
[13]	32	16	3.52	40,475	142,475
	16	29	5.22	22,415	117,009
[18]	32	13	5.46	80,187	437,825
	16	26	9.88	41,943	414,400
[19]	32	14	5.32	53,529	284,774
	16	27	9.18	28,695	263,422
[11]	32	6	3.24	83,050	269,082
	16	14	6.44	43,680	281,303
Fig. 2	32	16	3.84	39,668	152,328
	16	29	6.96	29,819	207,543
Fig. 3	32	14	6.3	12,773	80,474
	16	27	8.91	9,727	86,670

TABLE III
COMPARISON OF AREA-TIME COMPLEXITIES OF DIGIT-SERIAL KA-BASED MULTIPLIERS OVER $GF(2^{409})$ ON FPGA PLATFORM

Design	Digit-size	Latency	T_{TCD}	Area	Power	ADP	PDP
[14]^1	15	35	87.3	12,281	623	1,072,131	54,388
[15]^1	15	35	87.4	11,846	614	1,035,340	53,644
Fig. 2	16	29	137.8	3,900	245	537,459	33,763
Fig. 3	16	27	167.7	2,692	224	451,502	37,569

Unit for area: number of adaptive logic modules (ALMs).
Unit for T_{TCD}: ns.
Unit for power (dynamic power): mW@100MHz.
ADP=Area$\times T_{\text{TCD}}$.
PDP=Power$\times T_{\text{TCD}}$.
^{1}: The reported closest digit-size is 15 in [14] and [15].

ACKNOWLEDGMENTS

The research work of Jiafeng Xie is supported by the Villanova University Research Grant.

REFERENCES

[1] C. Andronic and D. Chiper, "A unified vlsi architecture for addition and multiplication in $GF(2^m)$," in *2015 ISSCS*. IEEE, 2015, pp. 1–4.

[2] Q. Shao, Z. Hu, S. N. Basha, Z. Zhang, Z. Wu, C.-Y. Lee, and J. Xie, "Low complexity implementation of unified systolic multipliers for nist pentanomials and trinomials over $GF(2^m)$," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 8, pp. 2455–2465, 2018.

[3] J. Xie and C.-Y. Lee, "LSM: Novel low-complexity unified systolic multiplier over binary extension field," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. ACM, 2019, pp. 343–346.

[4] J. Xie, C.-Y. Lee, P. K. Meher, and Z.-H. Mao, "Novel bit-parallel and digit-serial systolic finite field multipliers over $GF(2^m)$ based on reordered normal basis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 9, pp. 2119–2130, 2019.

[5] S. Namin, H. Wu, and M. Ahmadi, "Low-power design for a digit-serial polynomial basis finite field multiplier using factoring technique," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 441–449, 2017.

[6] J. L. Imaña, "High-speed polynomial basis multipliers over $GF(2^m)$ for special pentanomials," *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 63, no. 1, pp. 58–69, 2016.

[7] P. K. Meher and X. Lou, "Low-latency, low-area, and scalable systolic-like modular multipliers for $GF(2^m)$ based on irreducible all-one polynomials." *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 64, no. 2, pp. 399–408, 2017.

[8] C.-Y. Lee and P. K. Meher, "Subquadratic space-complexity digit-serial multipliers over $GF(2^m)$ using generalized (a,b)-way Karatsuba algorithm," *IEEE Transactions on Ciruits and Systems-I: Regular Papers*, vol. 62, no. 4, pp. 1091–1098, 2015.

[9] C. Lee and et al., "Low-complexity digit-serial and scalable SPB/GPB multipliers over large binary extension fields using (b, 2)-way Karatsuba decomposition," *IEEE Transactions on Ciruits and Systems-I: Regular Papers*, vol. 61, no. 11, pp. 3115–3124, 2014.

[10] P. Deschamps, J. L. Imaña, and D. Sutter, *Hardware implementation of finite-field arithmetic*. McGraw-Hill, Inc., 2009.

[11] C.-Y. Lee, P. K. Meher, and C.-H. Liu, "Area-delay efficient digit-serial multiplier based on k-partitioning scheme combined with TMVP block recombination approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, pp. 2413–2425, 2016.

[12] C.-H. Liu, C.-Y. Lee, and P. K. Meher, "Efficient digit-serial KA-based multiplier over binary extension fields using block recombination approach," *IEEE Transactions on Ciruits and Systems-I: Regular Papers*, vol. 62, no. 8, pp. 2044–2051, 2015.

[13] C.-Y. Lee, P. K. Meher, C.-C. Fan, and S.-M. Yuan, "Low-complexity digit-serial multiplier over $GF(2^m)$ based on efficient Toeplitz block toeplitz matrix–vector product decomposition," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 735–746, 2017.

[14] J. Xie and et al., "Efficient fpga implementation of low-complexity systolic karatsuba multiplier over $GF(2^m)$ based on NIST polynomials," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 7, pp. 1815–1825, 2017.

[15] J. Xie, P. K. Meher, X. Zhou, and C.-Y. Lee, "Low register-complexity systolic digit-serial multiplier over $GF(2^m)$," *IEEE Transactions on Multi-Scale Computing Systems*, no. 4, pp. 773–783, 2018.

[16] C. Lee and J. Xie, "BPB: A new finite field multiplier over $GF(2^m)$ based on bivariate polynomial basis," https://www.researchgate.net/publication/330505593-BPB-A-New-Finite-Field-Multiplier-over-GF-2-m-Based-on-Bivariate-Polynomial-Basis.

[17] C.-Y. Lee and J. Xie, "Low area-delay complexity digit-level parallel-in serial-out multiplier over $GF(2^m)$ based on overlap-free karatsuba algorithm," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 2018, pp. 187–194.

[18] C.-Y. Lee, "Concurrent error detection architectures for Gaussian normal basis multiplication over $GF(2^m)$," *Integration, the VLSI Journal*, vol. 43, no. 1, pp. 113–123, 2010.

[19] B. Uslu and S. S. Erdem, "Versatile digit serial multipliers for binary extension fields," *Computer & Electrical Engineering*, pp. 29–45, 2015.

[20] "Nangate standard cell library." [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib/