On the Complexity of Hybrid n-term Karatsuba Multiplier for Trinomials

Yin Li, Shantanu Sharma, Yu Zhang, Xingpo Ma, and Chuanda Qi

Abstract—The n-term Karatsuba algorithm (KA) is an extension of 2-term KA, which can obtain even fewer multiplications than the original one. The existing n-term Karatsuba hybrid $GF(2^m)$ multipliers rely on factorization of m or $m-1$, so that put a confinement to these schemes. In this contribution, we use a new decomposition $m = n\ell + r$, such that $0 \leq r < n$ and $0 \leq r < \ell$, and propose a novel n-term Karatsuba hybrid $GF(2^m)$ multiplier for an arbitrary irreducible trinomial $x^m + x^k + 1, m \geq 2k$. Combined with shifted polynomial basis, a new approach (other than Mastrovito approach) is introduced to exploit the spatial correlations among different subexpressions. We investigate the explicit space and time complexities, and discuss related upper and lower bounds. As a main contribution, the flexibilities of n, ℓ and r make our proposal approaching optimal for any given m. The space complexity can achieve to $m^2/2 + O(\sqrt{11}m^{3/2}/2)$, which roughly matches the best result of current hybrid multipliers for special trinomials. Meanwhile, its time complexity is slightly higher than the counterparts, but can be improved for a new class of trinomials. In particular, we demonstrate that the hybrid multiplier for $x^m + x^k + 1$, where k is approaching $\frac{m}{2}$, can achieve a better space-time trade-off than any other trinomials.

Index Terms—Hybrid multiplier, n-term Karatsuba algorithm, shifted polynomial basis, optimal, trinomials.

1 Introduction

We assume that the readers are familiar with the finite field $GF(2^m)$ and its arithmetic operations [1]. Such a field is a number system containing 2^m elements, where any field element is represented using m-bits in binary forms. Thus, it is particularly useful in coding theory and cryptography [4] that require efficient arithmetic algorithms and their related hardware architectures. Among the arithmetic operations defined in $GF(2^m)$, field multiplication is one of the most frequently desired operations, as other complex operations, e.g., exponentiation and division can be implemented by iterative multiplications. Therefore, it is essential to design a suitable $GF(2^m)$ multiplier under conditions of the different hardware resources.

Under polynomial basis (PB) representation [1], the field multiplication consists of a polynomial multiplication followed by a modular reduction. The PB $GF(2^m)$ multiplier aims to perform these operations efficiently. Typically, the multiplier efficiency is evaluated in terms of space and time complexities. The space complexity is expressed as the total number of required 2-input AND gates and 2-input XOR gates. The time complexity is defined as the total delay of the multiplier circuit, i.e., delays by AND and XOR gates, which is denoted by T_A and T_X, respectively. Broadly, we may classify bit-parallel $GF(2^m)$ multipliers into three different categories, as: (*i*) quadratic [13], [14], [22], [23], (*ii*) subquadratic [6], [7], and (*iii*) hybrid bit-parallel multipliers [9], [10], [11], [15], [21]. Quadratic multipliers normally utilize the trivial approach to implement the polynomial multiplication, while subquadratic or hybrid methods usually apply a certain divide-and-conquer algorithm, e.g., Karatsuba algorithm (KA) [2]. The main advantage of the sub-quadratic multipliers is that their space complexities are generally smaller than other two types of multipliers. Nevertheless, their time complexities are often more than quadratic or hybrid counterparts. Conversely, the hybrid multipliers can offer a trade-off between the time and space complexities [11]. Some of these schemes can save about $1/4$ logic gates, while the time complexities cost only one more T_X compared with the fastest quadratic multipliers [18], [30]. In these schemes, the KA is applied only once to compute the product of two m-degree polynomials.

Karatsuba algorithm is a classic divide-and-conquer algorithm, which can optimize polynomial multiplication by partitioning each polynomial into two halves and utilizing three sub-multiplications instead of four ones. This algorithm is usually denoted as 2-term Karatsuba algorithm. Besides the 2-term KA, there are several variations, e.g. generalized n-term KA ($n \geq 3$) introduced by Toom [3], Weimerskirch and Paar [7] and 4, 5 and 6-term of Karatsuba-like formulae introduced by Montgomery [5]. The former algorithm splits each polynomial into n parts and applies KA strategy for every two sub-polynomials. The latter ones introduced new formulae to minimize the number of sub-multiplications. Based on Montgomery's work, several combinations of these formulae resulted in remarkable improvements for higher degree polynomial multiplications [8]. Compared with 2-term KA, all these variations can obtain even fewer coefficient multiplications. However, these Karatsuba-like formulae usually contain

• *Manuscript received 30 June, 2019; accepted 27 November. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, including reprinting/republishing this material for advertising or promotional purposes, collecting new collected works for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The final published version of this paper may differ from this accepted version.*
Yin Li is with Dongguan University of Technology and Xinyang Normal University, P.R.China. Shantanu Sharma is with the University of California, Irvine, USA. Yu Zhang, Xingpo Ma and Chuanda Qi are with Xinyang Normal University, P.R.China. email: yunfeiyangli@gmail.com (Yin Li). *This work is supported by the National Natural Science Foundation of China (Grant no. 61402393, 61601396).*

complicated linear combinations of the split parts, which will yield extra gates delay for the bit-parallel multiplier. Conversely, Weimerskirch and Paar's approach [7] is more fit for constructing hybrid $GF(2^m)$ multipliers, as all the intermediate inputs can be obtained in one T_X delay [31]. We call this algorithm as n-term KA and use this notion thereafter.

Recently, Li et al. applied the n-term KA to a special class of trinomial $x^m + x^k + 1, m = nk$, and developed an efficient hybrid multiplier [30]. The optimal space complexity of the proposed multiplier is approximately $m^2/2 + O(m^{3/2})$, while its time delay matches the fastest Karatsuba-based multipliers known to date [18]. Nevertheless, such irreducible trinomials are not abundant, thereby puts a confinement to the application of this scheme. For example, the irreducible trinomials recommended by NIST [20] do not satisfy the previous precondition, so that the aforementioned multiplier is unapplicable. Park et al. [32] generalized above scheme and investigated the n-term Karatsuba hybrid multipliers for $x^m + x^k + 1$, where m satisfies $m = n\ell$ or $m = n\ell + 1$. A natural question is: can we loose the limitation for m, and develop an n-term KA based multiplier for an arbitrary irreducible trinomial?

This paper shows that, for a general irreducible trinomial $x^m + x^k + 1, m \geq 2k$, a hybrid n-term Karatsuba multiplier can be constructed by partitioning m as $m = n\ell + r$ with $0 \leq r < n, 0 \leq r < \ell$. Since the polynomial multiplication here is partitioned into several independent parts and computed in parallel, constructing Mastrovito matrices for all these parts becomes more complicated. Thus, we utilize an alternative approach to perform modular reduction and exploit spatial correlation among different subexpressions. We also use shifted polynomial basis (SPB) to optimize the modular reduction. The main architecture is described in detail, and the explicit space and time complexities are studied under different parameters n, ℓ and r.

Our contribution. We demonstrate that, the flexible choice of n, ℓ and r can make the n-term KA applicable to more general trinomials, especially for these recommended by NIST. The upper and lower bounds with respect to space and time complexities are evaluated. The optimal space complexity of our proposal is $m^2/2 + O(\sqrt{11}m^{3/2}/2)$, which roughly matches the best result of [30], [32]. The time complexity is slightly higher, but it can be improved for some special types of trinomials. Moreover, it is demonstrated that the hybrid multiplier for $x^m + x^{m/2} + 1$ can achieve a better space and time trade-off than any other trinomials.

Outline of the paper. Section 2 provides an overview of an n-term KA formula and relevant notions. Section 3 investigates the application of n-term KA for polynomial multiplication of arbitrary degrees and proposes a new bit-parallel multiplier architecture. Section 4 presents an analysis of our proposal and an study of the optimal parameters of n-term KA and irreducible trinomials.

Appendix. For easy reading, we put an small example, the detailed proofs of Lemma 1, Observation 1, and Propositions 1 and 3 in the Appendix.

2 PRELIMINARY

In this section, we briefly review some related notations and algorithms used throughout this paper.

2.1 Shifted Polynomial Basis

The shifted polynomial basis (SPB) was originally proposed by Fan and Dai [12] and it is a variation of the polynomial basis. Consider a binary extension field $GF(2^m)$ generated by an irreducible trinomial $f(x) = x^m + x^k + 1$. Let x be a root of $f(x)$, and the set $M = \{x^{m-1}, \cdots, x, 1\}$ constitute a polynomial basis (PB). Then, the SPB can be obtained by multiplying the set M by a certain exponentiation of x:

Definition 1 *[12] Let v be an integer and the ordered set $M = \{x^{m-1}, \cdots, x, 1\}$ be a polynomial basis of $GF(2^m)$ over $\mathbb{F}_2$. The ordered set $x^{-v}M := \{x^{i-v} | 0 \leq i \leq m-1\}$ is called the shifted polynomial basis(SPB) with respect to M.*

Under SPB representation, the field multiplication can be performed as: $C(x)x^{-v} = A(x)x^{-v} \cdot B(x)x^{-v} \bmod f(x)$. Please notice that the modular reduction under SPB is a little different with that under PB, where the range $[-v, m-v-1]$ is the rational term degree range. To distinguish with PB reduction, we call this operation as SPB reduction.

If the parameter v is properly selected, the SPB reduction is simpler than that PB reduction, especially, for irreducible trinomial or some special types of pentanomials [13]. Particularly, for trinomial $x^m + x^k + 1$, it is proved that the optimal value of v is k or $k-1$ [12]. In this paper, we choose that $v = k$ and use this denotation thereafter. Also, we utilize both SPB and PB reduction to our scheme, and the default modular reduction refers the SPB one without specification.

2.2 n-term Karatsuba Algorithm

Consider the simplest version of 2-term Karatsuba algorithm, where we assume two polynomials $A(x)$ and $B(x)$ of degree one, i.e., $A(x) = a_1x + a_0, B(x) = b_1x + b_0$. Then the polynomial multiplication $A(x)B(x)$ can be obtained in the following way:

$$AB = a_1b_1x^2 + ((a_0 + a_1)(b_0 + b_1) - a_1b_1 - a_0b_0)x + a_0b_0.$$

Above calculation requires only three multiplications, while the schoolbook method requires four multiplications. Besides 2-term KA, Weimerskirch and Paar [7] gave a generalized Karatsuba formulae, say an n-term KA, that is applicable for the polynomial multiplication of arbitrary degree. Assume that there are two n-term polynomials with $n-1$ degree over $\mathbb{F}_2$:

$$A(x) = \sum_{i=0}^{n-1} a_ix^i, \quad B(x) = \sum_{i=0}^{n-1} b_ix^i.$$

Then, we calculate intermediate values based on the coefficients. Compute for each $i = 0, \cdots, n-1$,

$$D_i = a_ib_i$$

Compute for each $i = 1, \cdots, 2n-3$ and for all s, t with $s + t = i$ and $n > s > t \geq 0$,

$$D_{s,t} = (a_s + a_t)(b_s + b_t)$$

Thus, the coefficients of $A(x)B(x) = \sum_{i=0}^{2n-2} c_i x^i$ can be computed as

$$\begin{aligned} &c_0 = D_0, \\ &c_{2n-2} = D_{n-1}, \\ &c_i = \begin{cases} \sum\limits_{\substack{s+t=i,\\ n>s>t\geq 0}} D_{s,t} + \sum\limits_{\substack{s+t=i,\\ n>s>t\geq 0}} (D_s + D_t) \ \text{(odd } i\text{)}, \\ \sum\limits_{\substack{s+t=i,\\ n>s>t\geq 0}} D_{s,t} + \sum\limits_{\substack{s+t=i,\\ n>s>t\geq 0}} (D_s + D_t) + D_{i/2} \ \text{(even } i\text{)}, \end{cases} \end{aligned}$$

where $i = 1, 2, \cdots, 2n-3$. Merging the similar items for $D_i, i = 0, 1, \cdots, n-1$, AB is rewritten as:

$$\begin{aligned} AB =& D_{n-1}(x^{2n-2} + \cdots + x^{n-1}) + D_{n-2}(x^{2n-3} + \cdots + x^{n-2}) \\ &+ \cdots + D_0(x^{n-1} + \cdots + 1) + \sum_{i=1}^{2n-3} \Big(\sum_{\substack{s+t=i,\\ n>s>t\geq 0}} D_{s,t} \Big) x^i. \end{aligned}$$

One can easily check that this formula costs about $O(n^2/2)$ coefficient multiplications and $O(5n^2/2)$ additions. Please note that the addition and subtraction are the same in $GF(2^m)$. Compared with the classic KA, the n-term KA saves more partial multiplications but costs more partial additions. It is noteworthy that the inputs of $D_{s,t}$ can be calculated using one addition, which make the computation of $D_{s,t}$ having only one more T_X in comparison with D_i. This characteristic is similar with 2-term KA; therefore, such an algorithm can be easily applied in developing bit-parallel multipliers. Besides this algorithm, Montgomery [5] and Fan [8] proposed more Karatsuba-like formulae. These formulae aim to decrease as many coefficient multiplications as possible. Nevertheless, their formulae contain more complicated linear combinations of subexpressions that require more gate delay for parallel implementation.

In the following section, we study the application of n-term KA in developing efficient bit-parallel multiplier for general irreducible trinomials.

3 Bit-parallel Multiplier using n-term Karatsuba Algorithm

In this section, we describe the bit-parallel $GF(2^m)$ multiplier architecture using n-term KA for general irreducible trinomials. We now give the explicit definition of the field $GF(2^m)$ and its element representation used in our multiplier. Provide that $f(x) = x^m + x^k + 1$ be an irreducible trinomial that defines the finite field $GF(2^m)$. Because the reciprocal polynomial $x^m + x^{m-k} + 1$ is also irreducible whenever $x^m + x^k + 1$ is irreducible, without loss of generality, we only consider the case of $m \geq 2k$. Let $A, B \in GF(2^m)$ are two arbitrary elements in PB representation, namely,

$$A = \sum_{i=0}^{m-1} a_i x^i, \ B = \sum_{i=0}^{m-1} b_i x^i.$$

Their SPB representation can be recognized as the PB representations multiplying x^{-k}. Analogous with PB multiplication, the SPB field multiplication consists of performing polynomial multiplication with parameter x^{-k} and then reducing the product modulo $f(x)$, i.e.,

$$\begin{aligned} Cx^{-k} =& x^{-k} \sum_{i=0}^{m-1} c_i x^i = Ax^{-k} \cdot Bx^{-k} \bmod f(x) \\ =& x^{-2k} \cdot \left(\sum_{i=0}^{m-1} a_i x^i \right) \cdot \left(\sum_{i=0}^{m-1} b_i x^i \right) \bmod f(x). \end{aligned}$$

To construct efficient bit-parallel multiplier for above expression, in the following, we first investigate the multiplication of two m-term polynomials using n-term KA ($m \geq n$). Then, the modular reductions for related results are considered. Accordingly, an efficient bit-parallel multiplier architecture is developed.

3.1 Polynomial multiplication using n-term Karatsuba algorithm

In order to apply n-term KA to the multiplication of two m-term polynomials $A \cdot B$, as presented in previous expression, we have to partition each polynomial into n parts. Notice that m is not always divisible by n. Therefore, we first decompose m as $m = n\ell + r$, where $0 \leq r < n$ and $0 \leq r < \ell$. Then, A, B can be partitioned into n parts with the former $n - r$ parts consisting of ℓ and the later r ones consisting of $\ell + 1$ bits. More explicitly,

$$\begin{aligned} A =& A_{n-1} x^{(n-1)\ell + r-1} + \cdots + A_{n-r+1} x^{(n-r+1)\ell+1} + A_{n-r} x^{(n-r)\ell} \\ &+ A_{n-r-1} x^{(n-r-1)\ell} + \cdots + A_1 x^{\ell} + A_0, \end{aligned}$$

and

$$\begin{aligned} B =& B_{n-1} x^{(n-1)\ell + r-1} + \cdots + B_{n-r+1} x^{(n-r+1)\ell+1} + B_{n-r} x^{(n-r)\ell} \\ &+ B_{n-r-1} x^{(n-r-1)\ell} + \cdots + B_1 x^{\ell} + B_0, \end{aligned}$$

where $A_i = \sum_{j=0}^{\ell-1} a_{j+i\ell} x^j, B_i = \sum_{j=0}^{\ell-1} b_{j+i\ell} x^j$, for $i = 0, 1, \cdots, n-r-1$, and $A_i = \sum_{j=0}^{\ell} a_{j+(\ell+1)i-n+r} x^j$, $B_i = \sum_{j=0}^{\ell} b_{j+(\ell+1)i-n+r} x^j$, for $i = n-r, \cdots, n-1$. Applying n-term KA stated in previous section to $A \cdot B$, we have the following proposition to illustrate the expansion of this polynomial multiplication.

Proposition 1 *Assume that A, B are defined as above, then the expansion of AB using n-term KA can be written as:*

$$\begin{aligned} AB =& \Big(A_{n-1} B_{n-1} x^{(n-1)\ell + r-1} + A_{n-2} B_{n-2} x^{(n-2)\ell + r-2} + \\ &\cdots + A_{n-r} B_{n-r} x^{(n-r)\ell} + \cdots + A_1 B_1 x^{\ell} + A_0 B_0 \Big) \cdot h(x) \\ &+ \sum_{i=1}^{2n-3} \Big(\sum_{\substack{s+t=i,\\ n>s>t\geq 0}} D_{s,t} x^{\delta_{s,t}} \Big) x^{i\ell} \end{aligned} \tag{1}$$

where $h(x) = x^{(n-1)\ell+r-1} + x^{(n-2)\ell+r-2} + \cdots + x^{(n-r)\ell} + \cdots + x^{\ell} + 1$ and $D_{s,t} = (A_s + A_t)(B_s + B_t)$ as well as

$$\delta_{s,t} = \begin{cases} s + t - 2(n-r), & \text{if } s > t > n-r, \\ s - (n-r), & \text{if } s > n-r, t \leq n-r, \\ 0, & \text{if } 0 < t < s \leq n-r. \end{cases} \tag{2}$$

The proof of this proposition can be found in Appendix B.1

Analogous to the approach present in [18], we can divide (1) into two parts and compute them independently, i.e.,

$$\begin{aligned} S_1 &= \left(\sum_{i=n-r+1}^{n-1} A_i B_i x^{i(\ell+1)-n+r} + \sum_{i=0}^{n-r} A_i B_i x^{i\ell} \right) h(x), \\ S_2 &= \sum_{i=1}^{2n-3} \Big(\sum_{\substack{s+t=i,\\ n>s>t\geq 0}} D_{s,t} x^{\delta_{s,t}} \Big) x^{i\ell}. \end{aligned}$$

Therefore, the SPB field multiplication is given by

$$Cx^{-k} = (S_1x^{-2k} + S_2x^{-2k}) \bmod x^m + x^k + 1$$

Next, in Section 3.2, we discuss the computation of S_1x^{-2k} and then S_2x^{-2k}, which is presented in Section 3.3.

3.2 Computation of $S_1x^{-2k} \bmod x^m + x^k + 1$

We note that the computation of $S_1x^{-2k} \bmod f(x)$ includes n sub-polynomial multiplications, the multiplication with $h(x)$ and the modular reductions. A straightforward way for this computation is utilizing Mastrivito approach, analogous to the authors did in [30], [32]. Nevertheless, we found that if the irreducible trinomial $f(x) = x^m + x^k + 1$ is not a specific one, e.g., $m = nk$, the associated Mastrovito matrix is far more complicated than that of $x^{nk} + x^k + 1$, which make it difficult to reuse logic gates and increase the overall space complexity. Therefore, we prefer an alternative approach that first computes the reduction of its subexpressions and then S_1x^{-2k}. Let

$$E(x) = \sum_{i=n-r+1}^{n-1} A_iB_ix^{i(\ell+1)-n+r} + \sum_{i=0}^{n-r} A_iB_ix^{i\ell}$$

So, $S_1x^{-2k} = E(x)h(x)x^{-2k}$. We first calculate $E(x)$ and then S_1x^{-2k} modulo $f(x)$. Based on the degrees of A_i, B_i, let $A_iB_i = (\sum_{j=0}^{\ell-1} a_{j+i\ell}x^j)\cdot(\sum_{j=0}^{\ell-1} b_{j+i\ell}x^j) = \sum_{j=0}^{2\ell-2} c_j^{(i)}x^j$, for $i = 0, 1, \cdots, n-r-1$, and

$$A_iB_i = (\sum_{j=0}^{\ell} a_{j+(\ell+1)i-n+r}x^j)\cdot(\sum_{j=0}^{\ell} b_{j+(\ell+1)i-n+r}x^j) = \sum_{j=0}^{2\ell} c_j^{(i)}x^j,$$

for $i = n-r, \cdots, n-1$. It is easy to check that $E(x)$ is of the degree $(n-1)\ell + r - 1 + 2\ell = m + \ell - 1$. Then, the coefficients of $E(x) = \sum_{i=0}^{m+\ell-1} e_ix^i$ are given by

$$e_i = \begin{cases} c_i^{(0)} & 0 \le i \le \ell-1, \\ c_i^{(0)} + c_{i-\ell}^{(1)} & \ell \le i \le 2\ell-2, \\ c_{i-\ell}^{(1)} & i = 2\ell-1, \\ c_{i-\ell}^{(1)} + c_{i-2\ell}^{(2)} & 2\ell \le i \le 3\ell-2, \\ \quad\vdots & \\ c_{i-(n-r-1)\ell}^{(n-r-1)} + c_{i-(n-r)\ell}^{(n-r)} & (n-r)\ell \le i \le (n-r+1)\ell-2, \\ c_{i-(n-r)\ell}^{(n-r)} & i = (n-r+1)\ell-1, (n-r+1)\ell \\ \quad\vdots & \\ c_{i-(n-2)\ell-r+2}^{(n-2)} + c_{i-(n-1)\ell-r+1}^{(n-1)} & (n-1)\ell + r - 1 \le i \le m-2, \\ c_{i-(n-1)\ell-r+1}^{(n-1)} & m-1 \le i \le m+\ell-1. \end{cases} \quad (3)$$

Recall that $\deg(E) = m + \ell - 1$. Obviously, there are only ℓ terms of $E(x)$ that needs to be reduced, if we perform the PB reduction with it. We partition $E(x)$ into $p_1x^m + p_0$, where $p_1(x) = \sum_{i=0}^{\ell-1} e_{i+m}x^i$ and $p_0(x) = \sum_{i=0}^{m-1} e_ix^i$. Then, we have

$$E(x) \bmod f(x) = p_1x^k + (p_1 + p_0).$$

Let $E'(x) = \sum_{i=0}^{m-1} e'_ix^i$ denote $p_1 + p_0$. The coefficients e'_is can be obtained by adding the ℓ most significant bits of $E(x)$ to its ℓ least significant bits, i.e.,

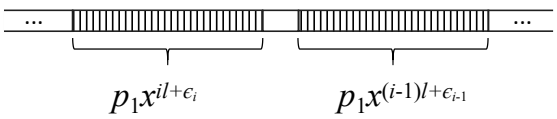

Fig. 1. Bit positions for $p_1x^{i\ell+\epsilon_i}, i = 1, 2, \cdots, n-1$.

$$e'_i = \begin{cases} c_i^{(0)} + c_{i+\ell+1}^{(n-1)} & 0 \le i \le \ell-1, \\ c_i^{(0)} + c_{i-\ell}^{(1)} & \ell \le i \le 2\ell-2, \\ c_{i-\ell}^{(1)} & i = 2\ell-1, \\ c_{i-\ell}^{(1)} + c_{i-2\ell}^{(2)} & 2\ell \le i \le 3\ell-2, \\ \quad\vdots & \\ c_{i-(n-r-1)\ell}^{(n-r-1)} + c_{i-(n-r)\ell}^{(n-r)} & (n-r)\ell \le i \le (n-r+1)\ell-2, \\ c_{i-(n-r)\ell}^{(n-r)} & i = (n-r+1)\ell-1, (n-r+1)\ell, \\ \quad\vdots & \\ c_{i-(n-2)\ell-r+2}^{(n-2)} + c_{i-(n-1)\ell-r+1}^{(n-1)} & (n-1)\ell + r - 1 \le i \le m-2, \\ c_{i-(n-1)\ell-r+1}^{(n-1)} & i = m-1. \end{cases} \quad (4)$$

In fact, one can calculate (4) instead of (3), and the computation of p_0 can be combined with that of (4). We give the details in Section 4.1.

After that, we then consider the SPB modular reduction of S_1x^{-2k}. Note that

$$\begin{aligned} S_1x^{-2k} \bmod f(x) &= E(x)h(x)x^{-2k} \bmod f(x) \\ &= [p_1x^k + (p_1 + p_0)]h(x)x^{-2k} \bmod f(x) \\ &= E'(x)h(x)x^{-2k} + p_1(x)h(x)x^{-k} \bmod f(x). \end{aligned}$$

In order to facilitate analysis, denoted by ϵ_i the extra term degrees in $h(x)$ except $i\ell, i = 0, 1, \cdots, n-1$, where $\epsilon_i = i - n + r$ if $i \ge n - r$ and 0 otherwise. Then,

$$E'(x)h(x)x^{-2k} = \sum_{i=0}^{n-1} E'(x)x^{-k}\cdot x^{i\ell+\epsilon_i-k},$$

$$p_1(x)h(x)x^{-k} = \sum_{i=0}^{n-1} p_1(x)x^{-k}\cdot x^{i\ell+\epsilon_i}.$$

On one hand, since p_1 consists of ℓ terms and $\epsilon_i \ge \epsilon_{i-1}$, there is no overlap between $p_1x^{i\ell+\epsilon_i}$ and $p_1x^{(i-1)\ell+\epsilon_{i-1}}$, for $i = 1, 2, \cdots, n-1$. Thus, the computation of $p_1(x)h(x)$ does not require any logic gates.Figure 1 depicts bit positions for these subexpressions. The axle parts represent the coefficients of $p_1(x)$, while the blank parts represent zeros.

Also, one can check that $\deg(p_1hx^{-k}) = (n-1)\ell + r - 1 + \ell - 1 - k = m - k - 2$, and all its term degrees are in the range $[-k, m-k-1]$. Therefore, under SPB representation, $p_1(x)h(x)x^{-k} \bmod f(x) = p_1(x)h(x)x^{-k}$ and no XOR gate is needed to compute this expression.

On the other hand, as $E'(x)$ is of degree $m-1$, $E'(x)x^{-k}$ can be viewed as an element of $GF(2^m)$ in SPB representation. The reduction of $E'(x)x^{-k}\cdot x^{i\ell+\epsilon_i-k}$ modulo $f(x)$ is equal to shifting $E'(x)x^{-k}$ by $i\ell + \epsilon_i - k$ bits in such a field. These operations depend on the magnitude relations between k and ℓ. Recall that $k \le m/2$, and $m = n\ell + r, n > r, \ell > r$. Two cases are considered:

1) $k \ge (n-1)\ell$
2) $k < (n-1)\ell$;

Particularly, if $n \geq 3$, we have

$$n\ell \geq 3\ell > 2\ell + r \Rightarrow (n-1)\ell > \ell + r$$
$$\Rightarrow 2(n-1)\ell > n\ell + r = m \Rightarrow (n-1)\ell > m/2 \geq k.$$

Therefore, the case of $k \geq (n-1)\ell$ happens only if $n = 2$. It is noteworthy that similar multiplier scheme using 2-term KA has already been studied in [18]. Thus, we only analyze the case of $k < (n-1)\ell$ in this study. The SPB reduction relies on the following formula:

$$\begin{cases} x^i = x^{m+i} + x^{i+k}, \text{ for } i = -2k, \cdots, -k-1; \\ x^i = x^{i-m} + x^{i-m+k}, \text{ for } i = m-k, \cdots, 2m-2k-2. \end{cases} \tag{5}$$

On top of that, we give a useful lemma.

Lemma 1 *Let $A(x) = \sum_{i=0}^{m-1} a_i x^{i-k}$ be an element of $GF(2^m)$ in SPB representation. Then, for an integer $-k \leq \Delta \leq m-k-1, \Delta \neq 0$, $A(x) \cdot x^{\Delta} \bmod x^m + x^k + 1$ can be expressed as*

$$\sum_{i=0}^{m-1} a_i x^{-k+(i+\Delta) \bmod m} + \sum_{i=m-\Delta}^{m-1} a_i x^{i+\Delta-m}, \text{ if } 1 \leq \Delta \leq m-k-1,$$
$$\sum_{i=0}^{m-1} a_i x^{-k+(i+\Delta) \bmod m} + \sum_{i=0}^{-\Delta-1} a_i x^{i+\Delta}, \text{ if } -k \leq \Delta < 0.$$

The proof of above lemma can be found in Appendix B.2. This lemma indicates that if we shift a $GF(2^m)$ element by Δ bits, the result equals a Δ-bit cyclic shift of its coefficients plus an extra expression of Δ bits.

Based on Lemma 1, we can perform the modular reduction with respect to $E'(x)x^{-k} \cdot x^{i\ell+\epsilon_i-k}$. Please notice that $i\ell + \epsilon_i - k$ here is equivalent to the integer Δ in Lemma 1. Let an integer t satisfy that $(t-1)\ell + \epsilon_{t-1} \leq k < t\ell + \epsilon_t$. Then, we have $i\ell + \epsilon_i - k \leq 0$, for $i = 0, 1, \cdots, t-1$ and $i\ell + \epsilon_i - k > 0$ for $i = t, \cdots, n-1$. The results of $E'(x)x^{i\ell+\epsilon_i-2k} \bmod f(x)$ are given by:

$$E'(x)x^{i\ell+\epsilon_i-2k} \bmod f(x) = \sum_{j=0}^{m-1} e'_j x^{-k+(j+\theta_i) \bmod m} + \sum_{j=0}^{-\theta_i-1} e'_j x^{j+\theta_i}, \tag{6}$$

for $i = 0, 1, \cdots, t-1$, and

$$E'(x)x^{i\ell+\epsilon_i-2k} \bmod f(x) = \sum_{j=0}^{m-1} e'_j x^{-k+(j+\theta_i) \bmod m} + \sum_{j=m-\theta_i}^{m-1} e'_j x^{j+\theta_i-m}, \tag{7}$$

for $i = t, \cdots, n-1$, where $\theta_i = i\ell + \epsilon_i - k$. Particularly, if $(t-1)\ell + \epsilon_{t-1} = k$, the corresponding expression

$$E'(x)x^{(t-1)\ell+\epsilon_{t-1}-2k} \bmod f(x) = E'(x)x^{(t-1)\ell+\epsilon_{t-1}-2k}$$

does not need any reduction. But it can be recognized as a special case of (6) with $\theta_{t-1} = 0$ and $\sum_{j=0}^{-\theta_{t-1}-1} d_j x^{j+\theta_{t-1}} = 0$. For simplicity, we do not discuss this case independently.

One can easily check that (6) and (7) consist of two subexpressions, in which the former contains m terms and the latter contains $|\theta_i|$ terms. Moreover, we note that the subexpressions $\sum_{j=0}^{-\theta_i-1} e'_j x^{j+\theta_i}$ $(i = 0, 1, \cdots, t-1)$ have all their term degrees smaller than 0, while $\sum_{j=m-\theta_i}^{m-1} e'_j x^{j+\theta_i-m}$ $(i = t, \cdots, n-1)$ have all their term degrees larger than 0. That is to say, there are no overlapped terms between these two kinds of subexpressions. We can add them without any logic gates.

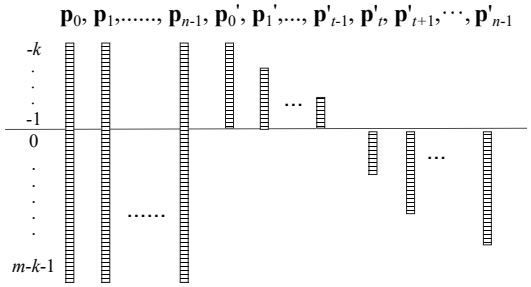

Fig. 2. Bit positions for all the subexpressions.

Figure 2 demonstrates the bit positions for these subexpressions. In this figure, the vectors $\mathbf{P}_i, \mathbf{P}'_i$ represent the coefficients vectors with respect to all the subexpressions in (6) and (7) and the axle parts express their nonzero coefficients. Recall that $p_1(x)h(x)x^{-k}$ is also needed to be added. In parallel implementation, it only needs $\lceil \log_2(n+1+\max\{t, n-t\}) \rceil T_X$ to add all these subexpressions together using a binary XOR tree. Moreover, as $t \geq 1$, we have $\lceil \log_2(n+1+\max\{t, n-t\}) \rceil \leq \lceil \log_2 2n \rceil$. Therefore, no more than $(1 + \lceil \log_2 n \rceil) T_X$ gates delays are needed for the modular reduction pertaining to $S_1 x^{-2k}$, after we finish computing $p_1 + p_0$ and p_1.

3.3 Computation of $S_2 x^{-2k} \bmod x^m + x^k + 1$

The computation of $S_2 x^{-2k}$ modulo $f(x)$ is different from that of $S_1 x^{-2k}$, as such an expression consists of $\binom{n}{2}$ different subexpressions $D_{s,t}x^{\delta}, (0 \leq t < s < n)$, each of which can be computed independently. One can see that A_i, B_i, for $i = 0, 1, \cdots, n-r-1$, are of degrees $\ell - 1$ and the rest of A_i, B_i are of degrees ℓ. Let $A_s + A_t = \sum_{i=0}^{\ell} u_i^{(s,t)} x^i, B_s + B_t = \sum_{i=0}^{\ell} v_i^{(s,t)} x^i$, for $0 \leq t < s, s \geq n-r$, and $A_s + A_t = \sum_{i=0}^{\ell-1} u_i^{(s,t)} x^i, B_s + B_t = \sum_{i=0}^{\ell-1} v_i^{(s,t)} x^i$, for $0 \leq t < s < n-r$. Then, we have

$$D_{s,t} = (\sum_{i=0}^{\ell-1} u_i^{(s,t)}) \cdot (\sum_{i=0}^{\ell-1} v_i^{(s,t)}) = \sum_{i=0}^{2\ell-2} d_i^{(s,t)} x^i, \tag{8}$$

if $0 \leq t < s < n-r$, and

$$D_{s,t} = (\sum_{i=0}^{\ell} u_i^{(s,t)}) \cdot (\sum_{i=0}^{\ell} v_i^{(s,t)}) = \sum_{i=0}^{2\ell} d_i^{(s,t)} x^i, \tag{9}$$

if $0 \leq t < s, s \geq n-r$. In order to perform modular reduction for $S_2 x^{-2k}$ efficiently, we apply a trick established in [29] to categorize all the $D_{s,t}$s, where the $D_{s,t}$s from the same category can be recognized as an integral to perform modular reduction. We first have the following proposition.

Proposition 2 *S_2 can be expressed as the plus of $g_1 x^{(2\lambda-1)\ell}$, $g_2 x^{(2\lambda-3)\ell}, \cdots, g_\lambda x^\ell$ for $\lambda = \frac{n}{2}$ (n is even) or $\lambda = \frac{n-1}{2}$ (n is odd), where*

$$g_1 = C_{n-1,n-2}x^{(n-2)\ell} + C_{n-1,n-3}x^{(n-3)\ell} + \cdots + C_{n-1,1}x^\ell + C_{n-1,0},$$
$$g_2 = C_{n-2,n-3}x^{(n-2)\ell} + C_{n-2,n-4}x^{(n-3)\ell} + \cdots + C_{n-2,0}x^\ell + C_{\frac{n}{2}-1,\frac{n}{2}-2},$$
$$g_3 = C_{n-3,n-4}x^{(n-2)\ell} + C_{n-3,n-5}x^{(n-3)\ell} + \cdots + C_{\frac{n}{2}-1,\frac{n}{2}-3}x^\ell + C_{\frac{n}{2}-1,\frac{n}{2}-4},$$
$$\vdots$$
$$g_{\frac{n}{2}} = C_{\frac{n}{2},\frac{n}{2}-1}x^{(n-2)\ell} + C_{\frac{n}{2},\frac{n}{2}-2}x^{(n-3)\ell} + \cdots + C_{2,0}x^\ell + C_{1,0},$$

or

$$g_1 = C_{n-1,n-2}x^{(n-1)\ell} + C_{n-1,n-3}x^{(n-2)\ell} + \cdots + C_{n-1,0}x^{\ell} + C_{\frac{n-1}{2},\frac{n-3}{2}},$$
$$g_2 = C_{n-2,n-3}x^{(n-1)\ell} + C_{n-2,n-4}x^{(n-2)\ell} + \cdots + C_{\frac{n-1}{2},\frac{n-5}{2}}x^{\ell} + C_{\frac{n-1}{2},\frac{n-7}{2}},$$
$$g_3 = C_{n-3,n-4}x^{(n-1)\ell} + C_{n-3,n-5}x^{(n-2)\ell} + \cdots + C_{\frac{n}{2}-1,\frac{n}{2}-3}x^{\ell} + C_{\frac{n}{2}-1,\frac{n}{2}-4},$$
$$\vdots$$
$$g_{\frac{n-1}{2}} = C_{\frac{n+1}{2},\frac{n-1}{2}}x^{(n-1)\ell} + C_{\frac{n+1}{2},\frac{n-3}{2}}x^{(n-2)\ell} + \cdots + C_{2,0}x^{\ell} + C_{1,0}.$$

Here, $C_{s,t} = D_{s,t} \cdot x^{\delta_{s,t}}$*, for* $n > s > t \geq 0$.

The proof of this proposition can be built using mathematical induction, which is nearly the same as the Proposition 1 in [29]. One just replaces $D_{s,t}$ by $D_{s,t} \cdot x^{\delta_{s,t}}$ in that proof and obtains the conclusion directly.

Based on Proposition 2,

$$S_2 x^{-2k} = g_1 x^{(2\lambda-1)\ell-2k} + g_2 x^{(2\lambda-3)\ell-2k} + \cdots + g_\lambda x^{\ell-2k}$$

Accordingly, its SPB reduction can also be expressed as a plus of these λ sub-expressions modulo $f(x)$. We can perform these modular reductions in parallel and then add the results together. The detailed computation for $S_2 x^{-2k} \bmod f(x)$ is presented as following steps:

(i) Perform bitwise addition $A_s + A_t, B_s + B_t, (n > s > t \geq 0)$ in parallel.
(ii) Classify the sub-expressions $D_{s,t}$ into λ parts according to Proposition 2 and compute these λ bigger expressions, i.e., $g_1, g_2, \cdots, g_\lambda$.
(iii) Perform reductions of $g_1 x^{(2\lambda-1)\ell-2k}, g_2 x^{(2\lambda-3)\ell-2k}, \cdots, g_\lambda x^{\ell-2k}$ modulo $f(x)$.
(iv) Add all these results using binary XOR tree to obtain the $S_2 x^{-2k} \bmod f(x)$.

Remark. In Step (i), there are $2 \cdot \binom{n}{2} = n(n-1)$ polynomial additions in all that need to be computed. All these additions can be performed in parallel, which costs one T_X delay. In Step (ii), we do not compute $D_{s,t}$ directly but λ integral expressions $g_1, \cdots, g_\lambda$. These computations are analogous to that of $E(x)$ in Section 3.2. The reduction of $S_2 x^{-2k}$ are performed in Step (iii) and Step (iv). Note that these steps can be computed jointly.

As polynomial additions in Step (i) are easy to implement, in the following, we mainly consider the computation of Step (ii)-(iv).

3.3.1 Step (ii)

Step (ii) consists of the computation of $g_1, g_2, \cdots, g_\lambda$, which are composed of $D_{s,t}$s. As mentioned in previous paragraphs, $D_{s,t}$s have different degrees. More explicitly, there are $\binom{n-r}{2}$ such $D_{s,t}$s of degrees $2\ell-2$ and $\binom{n}{2} - \binom{n-r}{2}$ $D_{s,t}$s of degrees 2ℓ. Therefore, according to Proposition 2, if n is even, $\lambda = \frac{n}{2}$, the degrees of $g_1, g_2, \cdots, g_{\frac{n}{2}}$ are at most $(n-2)\ell + 2\ell + 2r - 3 = m + r - 3$, if n is odd, $\lambda = \frac{n-1}{2}$, the degrees of $g_1, g_2, \cdots, g_{\frac{n-1}{2}}$ are at most $(n-1)\ell + 2\ell + 2r - 3 = m + \ell + r - 3$. We assume that $g_i = \sum_{j=0}^{m+r-3} h_j^{(i)} x^j$ if n is even, and $g_i = \sum_{j=0}^{m+\ell+r-3} h_j^{(i)} x^j$ if n is odd.

On top of that, $g_1, g_2, \cdots, g_\lambda$ have slightly different formulations as the $D_{s,t}$s in the same category may have different degrees and $\delta_{s,t}$ may also be different. We rewrite $D_{s,t}$ in a unified form: $D_{s,t} = \sum_{i=0}^{2\ell} d_i^{(s,t)} x^i$, with $d_{2\ell} = d_{2\ell-1} = 0$ if $0 \leq t < s < n - r$. According to the explicit formulation of g_i presented in Proposition 2, g_i consists of n (n is odd) or $n-1$ (n is even) subexpressions $D_{s,t}x^{\delta_{s,t}}$ and three arbitrary contiguous subexpressions in a same g_i have the following characteristic:

$$D_{s_1,t_1}x^{\delta_{s_1,t_1}+s\ell} + D_{s_2,t_2}x^{\delta_{s_2,t_2}+(s-1)\ell} + D_{s_3,t_3}x^{\delta_{s_3,t_3}+(s-2)\ell}$$

where $s_1 \geq s_2 \geq s_3$ and $s_1 + t_1 = s_2 + t_2 + 1 = s_2 + t_2 + 2$.

From (2), it is easy to obtain that $\delta_{s_1,t_1} \geq \delta_{s_2,t_2} \geq \delta_{s_3,t_3}$. One can check that only if $\delta_{s_1,t_1} = \delta_{s_2,t_2} = \delta_{s_3,t_3}$, the corresponding coefficients of g_i are overlapped by these three subexpressions. Part of its coefficients are given by:

$$h_j^{(i)} = \begin{cases} \vdots & \vdots \\ d_{j-(s-3)\ell-\delta}^{(s_3,t_3)} + d_{j-(s-2)\ell-\delta}^{(s_2,t_2)}, & (s-2)\ell + \delta \leq j \leq (s-1)\ell + \delta - 1, \\ d_{2\ell}^{(s_3,t_3)} + d_{\ell}^{(s_2,t_2)} + d_0^{(s_1,t_1)}, & j = (s-1)\ell + \delta, \\ d_{j-(s-2)\ell-\delta}^{(s_2,t_2)} + d_{j-(s-1)\ell-\delta}^{(s_1,t_1)}, & (s-1)\ell + \delta + 1 \leq j \leq s\ell + \delta - 1, \\ \vdots & \vdots \end{cases}$$

where $\delta = \delta_{s_1,t_1} = \delta_{s_2,t_2} = \delta_{s_3,t_3}$.We note that in this case, only when $d_{2\ell}^{(s_3,t_3)} \neq 0$, $h_{(s-1)\ell+\delta}^{(i)}$ is a plus of three terms. Otherwise, there is no coefficient of g_i obtained by a plus of three terms. Plug (8) and (9) into above formula, it is easy to check that $h_{(s-1)\ell+\delta}^{(i)}$ contains $\ell + 3$ terms of $u_i^{(s,t)} \cdot v_i^{(s,t)}$, which leads to at most $\lceil \log_2(\ell+3) \rceil T_X$ delays using binary XOR tree. Also notice that one T_A is needed to calculate the coefficient multiplication related to $D_{s,t}$. We immediately obtain that all g_is can be implemented in parallel using $T_A + \lceil \log_2(\ell+3) \rceil T_X$ gates delay.

3.3.2 Step (iii) and (iv)

Now, we consider the computations of Step (iii) and (iv). Firstly, we have the following observation.

Observation 1 The modular reduction of $g_1 x^{(2\lambda-1)\ell-2k}$, $g_2 x^{(2\lambda-3)\ell-2k}$, $\cdots, g_\lambda x^{\ell-2k}$ by $f(x)$ only require one reduction step.

The proof of this observation is given in Appendix B.3. We then investigate the computation of Step (iii). For simplicity, let $\Delta_i = (2\lambda - 2i + 1)\ell - k$, $i = 1, 2, \cdots, \lambda$, then $g_1 x^{(2\lambda-1)\ell-2k}, g_2 x^{(2\lambda-3)\ell-2k}, \cdots, g_\lambda x^{\ell-2k}$ can be rewritten in a unified form, i.e.,

$$g_i x^{\Delta_i - k}, i = 1, 2, \cdots, \lambda$$

Please notice that the explicit reduction formulations of $g_i x^{\Delta_i - k}$ modulo $f(x)$ depend on the choice of n, ℓ and k. According to the previous statement, it is clear that $n \geq 2$ and thus $\ell \leq m/2$. We also have $0 < k \leq m/2$. But, the magnitude relations of these parameters are uncertain, which highly influence the application of the reduction rule. For example, if $\ell > k$, we have $\ell - 2k > -k$. All the terms of $g_i x^{\Delta_i - k}$ have their degrees larger than $-k$. We only need to reduce the terms whose degrees are greater than $m - k - 1$. Therefore, to investigate the modular reduction details, six cases are considered:

1) n is even, $\ell < k, (n-1)\ell \leq k$;
2) n is even, $\ell < k, (n-1)\ell > k$;
3) n is even, $\ell \geq k$;
4) n is odd, $\ell < k, (n-2)\ell \leq k$;
5) n is odd, $\ell < k, (n-2)\ell > k$;
6) n is odd, $\ell \geq k$.

As described in Section 3.2, Case 1 happens only if $n = 2$, which has already been studied in [18]; thus, we only analyze the rest of the cases, separately.

Since the degrees of g_i are at most $m + r - 3$ (n is even) or $m + \ell + r - 3$ (n is odd), we partition g_i into two parts accordingly, i.e., $g_i = p_1^{(i)}x^m + p_0^{(i)}$, for $i = 1, 2, \cdots, \lambda$, where the first part consists of $r - 2$ (or $\ell + r - 2$) terms and latter one consists of m terms. We directly have

$$g_i \bmod f(x) = p_1^{(i)}(x^k + 1) + p_0^{(i)}$$

Thus, the modular reductions with respect to $g_i x^{\Delta_i - k}$ can be expressed as the reduction with respect to $p_1^{(i)}, p_0^{(i)}$ multiplying certain exponent of x. More explicitly,

$$\begin{aligned} g_i x^{\Delta_i - k} \bmod f(x) = \Big(p_1^{(i)} + p_1^{(i)}x^{-k} + \\ p_0^{(i)}x^{-k}\Big) x^{\Delta_i} \bmod f(x), \end{aligned} \tag{10}$$

$i = 1, 2, \cdots, \lambda$. It is clear that the expressions $p_1^{(i)}$, $p_1^{(i)}x^{-k}$ and $p_0^{(i)}x^{-k}$ have all their term degrees in the range $[-k, m-k-1]$. Therefore, the modular reductions of $g_i x^{\Delta_i - k}$ will also utilize Lemma 1. Take into account this lemma, we have following proposition.

Proposition 3 *Step (iii) and (iv) can be calculated jointly within at most* $\lceil \log_2(n+2) \rceil T_X$ *delay.*

Proof Obviously, Step (iii) and (iv) actually compute $\sum_{i=1}^{\lambda} g_i x^{(2\lambda - 2i + 1)\ell - 2k} \bmod f(x)$, which consists of polynomial modular reductions and additions. Without loss of generality, we only analyze Case 2 here, the proof for the rest of cases are available in Appendix B.4.

In this case, recall that $\Delta_i = (n - 2i + 1)\ell - k, i = 1, 2, \cdots, \frac{n}{2}$. Since $\ell < k, (n-1)\ell > k$, one can check that some of Δ_is are greater than 0 and others are less than 0, which will lead to different reduction formulae according to Lemma 1.

Let an odd integer $t \geq 1$ satisfy that $t\ell \leq k, (t+2)\ell > k$. Then, we have $\Delta_i > 0$, for $i = 1, 2, \cdots, \frac{n-t-1}{2}$ and $\Delta_i \leq 0$ for $i = \frac{n-t+1}{2}, \cdots, \frac{n}{2}$. Now we investigate the detailed modular reduction of (10). Note that $p_1^{(i)} = \sum_{j=0}^{r-3} h_{m+j}^{(i)} x^j$ and $p_0^{(i)} = \sum_{j=0}^{m-1} h_j^{(i)} x^j$ here. Firstly, the modular reduction of $p_0^{(i)} x^{\Delta_i - k}$ can be obtained as follows:

$$\begin{aligned} p_0^{(i)} x^{\Delta_i - k} \bmod f(x) = \sum_{j=0}^{m-1} h_j^{(i)} x^{-k + (j + \Delta_i) \bmod m} \\ + \sum_{j=m-\Delta_i}^{m-1} h_j^{(i)} x^{j + \Delta_i - m}, \end{aligned} \tag{11}$$

for $i = 1, 2, \cdots, \frac{n-t-1}{2}$, and

$$\begin{aligned} p_0^{(i)} x^{\Delta_i - k} \bmod f(x) = \sum_{j=0}^{m-1} h_j^{(i)} x^{-k + (j + \Delta_i) \bmod m} \\ + \sum_{j=0}^{-\Delta_i - 1} h_j^{(i)} x^{j + \Delta_i}, \end{aligned} \tag{12}$$

for $i = \frac{n-t+1}{2}, \cdots, \frac{n}{2}$.

Then, consider the reduction of $p_1^{(i)} x^{\Delta_i} + p_1^{(i)} x^{\Delta_i - k}$. We know that the maxium degree of $p_1^{(i)}$ is $r - 3$ and $\max \Delta_i = (n-1)\ell - k < m - k - \ell$. Thus, it is easy to check that the degrees of $p_1^{(i)} x^{\Delta_i}$ ($i = 1, \cdots, \frac{n}{2}$) are all in the range $[-k, m-k-1]$, which need no reduction. That is to say,

$$\sum_{i=1}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i} \bmod f(x) = \sum_{i=1}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i}. \tag{13}$$

However, as $t\ell < k$, $p_1^{(i)} x^{\Delta_i - k}, i = \frac{n-t+1}{2}, \cdots, \frac{n}{2}$ have some term degrees less than $-k$ and thus need reduction by $f(x)$. Specifically, we note that $\deg(p_1^{(\frac{n-t+1}{2})}) \leq r - 3$. It is possible that $t\ell < k$ but $t\ell + r - 3 \geq k$, which indicates that a part of $p_1^{(\frac{n-t+1}{2})} x^{t\ell - 2k}$does not need further reduction. Therefore, the explicit reduction formulae are given by

$$p_1^{(i)} x^{\Delta_i - k} \bmod f(x) = p_1^{(i)} x^{m + \Delta_i - k} + p_1^{(i)} x^{\Delta_i}, \tag{14}$$

for $i = \frac{n-t+3}{2}, \cdots, \frac{n}{2}$. And,

$$\begin{aligned} & p_1^{(\frac{n-t+1}{2})} x^{t\ell - 2k} \bmod f(x) \\ &= \left(p_{1,1}^{(\frac{n-t+1}{2})} x^{k - t\ell} + p_{1,2}^{(\frac{n-t+1}{2})}\right) x^{t\ell - 2k} \bmod f(x) \\ &= p_{1,1}^{(\frac{n-t+1}{2})} x^{-k} + p_{1,2}^{(\frac{n-t+1}{2})} (x^{m + t\ell - 2k} + x^{t\ell - k}). \end{aligned} \tag{15}$$

Here, $p_{1,1}^{(\frac{n-t+1}{2})}$ consists of at most $r - 2 - (k - t\ell)$ bits and $p_{1,2}^{(\frac{n-t+1}{2})}$ consists of at most $k - t\ell$ bits. [1].

Moreover, note that $\Delta_i - \Delta_{i+1} = 2\ell$ for $i = 1, 2, \cdots, \frac{n}{2} - 1$ and each $p_1^{(i)}$ consists of at most $r - 2$ terms. There are no overlapped terms among $p_1^{(1)} x^{\Delta_1}, p_1^{(2)} x^{\Delta_2}, \cdots, p_1^{(\frac{n}{2})} x^{\Delta_{\frac{n}{2}}}$, so we can add them without any logic gates. Similar thing also happens among $p_1^{(i)} x^{m + \Delta_i - k}$, ($i = \frac{n-t+3}{2}, \cdots, \frac{n}{2}$), and $p_1^{(i)} x^{\Delta_i - k}$, ($i = 1, 2, \cdots, \frac{n-t-1}{2}$). By combining the same sub-expressions and swapping some parts of (13), (14) and (15), the result of $\sum_{i=1}^{\frac{n}{2}} (p_1^{(i)} + p_1^{(i)} x^{-k}) x^{\Delta_i}$ modulo $f(x)$ can be written as two independent expressions:

$$\begin{aligned} & \sum_{i=1}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i} + \sum_{i=\frac{n-t+3}{2}}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i} + p_{1,1}^{(\frac{n-t+1}{2})} x^{-k} + p_{1,2}^{(\frac{n-t+1}{2})} x^{t\ell - k} \\ &= \sum_{i=1}^{\frac{n-t-1}{2}} p_1^{(i)} x^{\Delta_i} + p_{1,1}^{(\frac{n-t+1}{2})} (x^{t\ell - k} + x^{-k}), \end{aligned} \tag{16}$$

$$\sum_{i=\frac{n-t+3}{2}}^{\frac{n}{2}} p_1^{(i)} x^{m + \Delta_i - k} + \sum_{i=1}^{\frac{n-t-1}{2}} p_1^{(i)} x^{\Delta_i - k} + p_{1,2}^{(\frac{n-t+1}{2})} x^{m + t\ell - 2k}, \tag{17}$$

each of which consists of sub-expressions that have no overlapped terms.

Finally, we add all the modular reduction results included in (11), (12), (16) and (17) to obtain $S_2 x^{-2k} \bmod f(x)$. Specifically, we note that the subexpression $\sum_{j=m-\Delta_i}^{m-1} h_j^{(i)} x^{j + \Delta_i - m}$ in (11) does not overlap with $\sum_{j=0}^{-\Delta_i - 1} h_j^{(i)} x^{j + \Delta_i}$ in (12), so that every two of such expressions can be concatenated together. This case is similar with what happened in Figure 2. As a result, we only need to add $\frac{n}{2} + 2 + \max\{\frac{n-t-1}{2}, \frac{t+1}{2}\}$ combined expressions using binary XOR tree, which requires $\lceil \log_2(\frac{n}{2} + 2 + \max\{\frac{n-t-1}{2}, \frac{t+1}{2}\}) \rceil T_X \leq \lceil \log_2(n+2) \rceil T_X$ delay in parallel. Then, we conclude the proposition. □

1. If $t\ell + r - 3 < k$, we have $p_{1,1}^{(\frac{n-t+1}{2})} = 0$ and $p_{1,2}^{(\frac{n-t+1}{2})} = p_1^{(\frac{n-t+1}{2})}$, which does not influence the result.

TABLE 1
Organization of S_1, S_2 and the computation sequence.

I: $S_1x^{-6} \bmod x^7+x^3+1$	
(i)	$U:(A_2B_2x^4+A_1B_1x^2+A_0B_0) \bmod x^7+x^3+1$
(ii)	$U\cdot(x^4+x^2+1)\cdot x^{-6} \bmod x^7+x^3+1$
II: $S_2x^{-6} \bmod x^7+x^3+1$	
(i)	$A_2+A_1, B_2+B_1, A_2+A_0, B_2+B_0, A_1+A_0, B_1+B_0$
(ii)	$(D_{2,1}x^6+D_{2,0}x^4+D_{1,0}x^2)\cdot x^{-6} \bmod x^7+x^3+1$

Special cases. If $0 \le r \le 2$ and n is even, we can see that the degrees of g_is are less than $m-1$. It is not necessary to partition g_i into two parts and $g_i = p_0^{(i)}, p_1^{(i)} = 0$. In this case, the modular reduction of $g_i x^{\Delta_i - k}$ can be obtained using (11) and (12), which also obeys previous proposition.

3.4 A small example of n-term Karatsuba Multiplier

We give a small example to illustrate the basic idea of our scheme. Consider an irreducible trinomial x^7+x^3+1 that defines a finite field $GF(2^7)$ using SPB representation. That is to say, the group $\{x^{-3}, x^{-2}, \cdots, x^2, x^3\}$ constitutes the basis. Assume that $A, B \in GF(2^7)$ are two arbitrary elements, where

$$A = a_6x^3+a_5x^2+a_4x+a_3+a_2x^{-1}+a_1x^{-2}+a_0x^{-3}$$
$$B = b_6x^3+b_5x^2+b_4x+b_3+b_2x^{-1}+b_1x^{-2}+b_0x^{-3}$$

We now consider their SPB multiplication. Without loss of generality, we apply a 3-term KA to the polynomial multiplication. It is clear that $7 = 3\times 2+1$, we then have $n=3, \ell=2, r=1$ and r satisfies the condition $0 \le r < n, 0 \le r < \ell$. Accordingly, we partition A, B into three parts, with the first part containing three coefficients and other parts containing two coefficients.

$$\begin{aligned} A &= A_2x + A_1x^{-1} + A_0x^{-3} \\ &= \underbrace{(a_6x^2+a_5x+a_4)}_{A_2}x + \underbrace{(a_3x+a_2)}_{A_1}x^{-1} + \underbrace{(a_1x+a_0)}_{A_0}x^{-3}, \\ B &= B_2x + B_1x^{-1} + B_0x^{-3} \\ &= \underbrace{(b_6x^2+b_5x+b_4)}_{B_2}x + \underbrace{(b_3x+b_2)}_{B_1}x^{-1} + \underbrace{(b_1x+b_0)}_{B_0}x^{-3}. \end{aligned}$$

Then, the polynomial multiplication $A \times B$ can be performed using a 3-term KA, as follows:

$$\begin{aligned} AB &= (A_2x+A_1x^{-1}+A_0x^{-3})\times(B_2x+B_1x^{-1}+B_0x^{-3}) \\ &= (A_2B_2x^8+(A_1B_2+A_2B_1)x^6+(A_0B_2+A_1B_1 \\ &\quad +A_2B_0)x^4+(A_0B_1+A_1B_0)x^2+A_0B_0)x^{-6} \\ &= \Big[\underbrace{A_2B_2(x^8+x^6+x^4)+A_1B_1(x^6+x^4+x^2)}_{+A_0B_0(x^4+x^2+1)_{S_1}} + \overbrace{(A_2+A_1)(B_2+B_1)}^{D_{2,1}}x^6 \\ &\quad + \overbrace{(A_2+A_0)(B_2+B_0)}^{D_{2,0}}x^4 + \overbrace{(A_1+A_0)(B_1+B_0)}^{D_{1,0}}x^2{}_{S_2}\Big]x^{-6}. \end{aligned}$$

Assume that $D_{s,t}$ denote $(A_s+A_t)(B_s+B_t)$, for $0 \le t < s < 2$. Now, we split the above expression in the square bracket into S_1 and S_2 and compute them in parallel, refer to Table 1. The computation details for the steps of Table 1 are available in the appendix A.

After obtaining the results of S_1x^{-6} and S_2x^{-6} modulo x^7+x^3+1, we only need to add them to obtain the ultimate result of $AB \bmod x^7+x^3+1 = S_1x^{-6}+S_2x^{-6}$.

4 Complexity Analysis

Based on previous description, in this section, we analyze the space and time complexities pertaining to S_1x^{-2k} and S_2x^{-2k} modulo $f(x)$.

4.1 Space and time complexity of $S_1x^{-2k} \bmod f(x)$

As presented in section 3.2, the computation of S_1x^{-2k} modulo $f(x)$ consists of computation of p_1, p_1+p_0 following a modular multiplication by $h(x)x^{-2k}$. We first investigate the complexity of p_1 and $E' = p_1+p_0$. From (3) and (4), we can see that the coefficients of p_1 and p_1+p_0 are composed of $c_j^{(i)}$ $(i=0,1,\cdots,n-1)$, where

$$c_j^{(i)} = \begin{cases} \sum_{t=0}^{j} a_{t+i\ell}b_{t-j+i\ell} & 0 \le t \le \ell-1, \\ \sum_{t=j-\ell+1}^{\ell-1} a_{t+i\ell}b_{t-j+i\ell} & \ell \le t \le 2\ell-2, \end{cases}$$

for $i=0,1,2,\cdots,n-r-1$, and

$$c_j^{(i)} = \begin{cases} \sum_{t=0}^{j} a_{t+(\ell+1)i-n+r}b_{t-j+(\ell+1)i-n+r} & 0 \le t \le \ell, \\ \sum_{t=j-\ell}^{\ell} a_{t+(\ell+1)i-n+r}b_{t-j+(\ell+1)i-n+r} & \ell+1 \le t \le 2\ell, \end{cases}$$

for $i = n-r, \cdots, n-1$. Combine the above expressions with (3) and (4), it is easy to check that each coefficient e_i and e'_i are composed of at most $\ell+1$ coefficient products of $A_iB_i, i=0,1,\cdots,n-1$. We immediately conclude that p_1+p_0 and p_1 can be computed in $T_A+\lceil\log_2(\ell+1)\rceil T_X$ delay. Table 2 presents the gate count and time delay for the implementation of each coefficient of p_1+p_0. Furthermore, notice that

$$p_1(x) = \sum_{i=0}^{\ell-1} e_{i+m}x^i = \sum_{i=0}^{\ell-1} c_{i+\ell+1}^{(n-1)}x^i.$$

It is obvious that $E' = p_1+p_0$ contains all the terms that are included in p_1. Therefore, no AND gates are needed to compute p_1, and some XOR gates can also be saved using a so-called binary tree *sub-expression sharing* [17], [18]. The authors found that if two binary XOR trees share k common items, $k-W(k)$ XOR gates can be saved, where $W(k)$ is the Hamming weight of the binary representation of k. Here, the coefficient $e_{i+m}(i=0,1,\cdots,\ell-1)$ of p_1 consists of $i+1$ items a_ib_j and shares $i+1$ items with those coefficients of E', only $i-(i+1-W(i+1)) = W(i+1)-1$ XOR gates are needed. Thus, it totally requires $\sum_{i=1}^{\ell} W(i)-\ell$ XOR gates in all to compute p_1.

We then investigate the complexity of $E'(x)h(x)x^{-2k} + p_1h(x)x^{-k}$ modulo $f(x)$. As shown in Section 3.2, we only need to add at most $2n+1$ expressions to obtain the result. Particularly, note that vectors $\mathbf{P}_0, \cdots, \mathbf{P}_{n-1}$ consist of m bits, while $\mathbf{P}'_0, \cdots, \mathbf{P}'_{n-1}$ consist of $|\theta_i|$ bits. Also, $p_1h(x)x^{-k}$ contains at most $n\ell$ nonzero items. Thus, the number of required XOR gates is

$$n\ell + (n-1)m + \sum_{i=0}^{n-1} |i\ell+\epsilon_i-k|$$

Table 3 summarizes the space and time complexity for every step of $S_1x^{-2k} \bmod f(x)$.

TABLE 2
The computation complexity of e'_i

e'_i	#AND	#XOR	Delay
$e'_0 = c_0^{(0)} + c_{\ell+1}^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e'_{\ell-1} = c_{\ell-1}^{(0)} + c_{2\ell}^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$
$e'_\ell = c_\ell^{(0)} + c_0^{(1)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil) T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e'_{2\ell-1} = c_{\ell-1}^{(1)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil) T_X$
$e'_{2\ell} = c_\ell^{(1)} + c_0^{(2)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil) T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e'_{(n-r+1)\ell-1} = c_{\ell-1}^{(n-r)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil) T_X$
$e'_{(n-r+1)\ell} = c_\ell^{(n-r)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e'_{(n-1)\ell+r-1} = c_{\ell+1}^{(n-2)} + c_0^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e'_{m-1} = c_\ell^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$
Total	$(n-r)\ell^2 + r(\ell+1)^2$	$(n-r)\ell(\ell-1) + r\ell(\ell+1)$	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$

TABLE 3
Space and time complexities of $S_1 x^{-2k} \bmod f(x)$.

Operation	# AND	#XOR	Delay
$E' = p_1 + p_0$	$n\ell^2 + 2\ell r + r$	$n\ell^2 + 2\ell r - n\ell$	$T_A +$
p_1	-	$\sum_{i=1}^{\ell} W(i) - \ell$	$\lceil \log_2(\ell+1) \rceil T_X$
$S_1 \bmod f(x)$	-	$n\ell + (n-1)m + \sum_{i=0}^{n-1} \lvert\theta_i\rvert$	$\leq \lceil \log_2 2n \rceil T_X$
where $\theta_i = i\ell + \epsilon_i - k$, $\epsilon_i = i - n + r$ for $i = n-r, \cdots, n-1$, $\epsilon_i = 0$ for $i = 0, 1, \cdots n-r-1$			

TABLE 4
Number of bits in (11), (12), (16) and (17).

Formulae	number of bits
(11)	$m + \Delta_i, i = 1, 2, \cdots, \frac{n-t-1}{2}$
(12)	$m + \lvert\Delta_i\rvert, i = \frac{n-t+1}{2}, \cdots, \frac{n}{2}$
(16)	$\frac{n-t-1}{2} \cdot (r-2) + 2(r-2-(k-t\ell))$
(17)	$(\frac{n}{2} - 1) \cdot (r-2) + (k - t\ell)$

4.2 Space and time complexity of $S_2 x^{-2k} \bmod f(x)$

Now, we discuss the complexity of $S_2 x^{-2k} \bmod f(x)$ step by step. Firstly, based on the description in Section 3.3, it is easy to check that $A_s + A_t, B_s + B_t$ for $0 \leq t < n-r$ requires ℓ XOR gates each, while each of $A_s + A_t, B_s + B_t$ for $s > t \geq n-r$ costs $\ell+1$ XOR gates. Since there are $\binom{n}{2}$ different such expressions, these additions totally require

$$2 \cdot \left(\frac{r(r-1)}{2}(\ell+1) + \left(\frac{n(n-1)}{2} - \frac{r(r-1)}{2} \right) \ell \right) = n^2\ell + r^2 - m$$

XOR gates for the pre-computation of all the $A_s + A_t, B_s + B_t$.

Secondly, the computation of $g_1, g_2, \cdots, g_\lambda$ contains the computation of $D_{s,t}$s and the additions among $D_{s,t}$s in the same category. Recall that $D_{s,t}$s have different degrees. Thus, the computation of one $D_{s,t}$ costs ℓ^2 AND gates plus $(\ell-1)^2$ XOR gates if its degree is $2\ell-2$, otherwise it costs $(\ell+1)^2$ AND and ℓ^2 XOR gates.In addition, one can check that when adding $D_{s,t}$s to obtain g_i, only the ℓ least significant bits and ℓ most significant bits of g_i do not need additions. We also note that every three contiguous $D_{s,t}$s may overlap each other, as presented in Section 3.3.1. In this case, the coefficient of g_i, i.e., $h_j^{(i)}$ a plus of three terms, which require one extra XOR gate. Fortunately, it happens only if $D_{s,t}$ is of degree ℓ. There are $\binom{n}{2} - \binom{n-r}{2} = nr - (r^2+r)/2$ such $D_{s,t}$s, each of which requires one extra XOR gates. Hence, the additions among $D_{s,t}$s in these categories totally require $\frac{n}{2} \cdot (m + r - 2 - 2\ell)$ (even n) or $\frac{n-1}{2} \cdot (m + r - 2 - \ell)$ (odd n) XOR gates plus $nr - (r^2+r)/2$ extra XOR gates.

In the end, as mentioned in Section 3.3, we need to add the modular results presented in (11), (12), (16) and (17) to obtain the final result. The number of required XOR gates depends on these formulations. For example, Table 4 presents the number of bits included in (11), (12), (16) and (17) for Case 2. It requires at most $m(\frac{n}{2} - 1) + (n - \frac{t-1}{2})(r-2) + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert - (k - t\ell)$ XOR gates to add these expressions. Specifically, recall that if $0 \leq r \leq 2$ and n is even, we only need to add (11), (12). As in this sub-case, all g_i contains at most m bits and no partition is needed.

Space and time complexities for each steps are given in Table 5, when $r > 2$. Also, if $0 \leq r \leq 2$, the number of XOR gates of (iii)-(iv) in Case 2 and 3 does not include the expressions about $r-2$ and ε, as they are equal to zero, here. Accordingly, the delays for these steps are $\lceil \log_2(\frac{n}{2} + \max\{\frac{n-t-1}{2}, \frac{t+1}{2}\}) \rceil T_X$ and $\lceil \log_2 n \rceil T_X$, respectively.

4.3 Total complexity and discussion

As mentioned in previous sections, $S_1 x^{-2k} \bmod f(x)$ and $S_2 x^{-2k} \bmod f(x)$ are computed in parallel and the overall delay is equal to the larger gates delays of either $S_1 x^{-2k} \bmod f(x)$ or $S_2 x^{-2k} \bmod f(x)$. From Tables 3 and 5, it is clear that the delay of $S_2 x^{-2k} \bmod f(x)$ is slightly higher. Thus, the overall time delay for parallel implementation of $S_1 x^{-2k}, S_2 x^{-2k}$ modulo $f(x)$ is $T_A + (1 + \lceil \log_2(\ell+3) \rceil + \lceil \log_2(n+2) \rceil) T_X$. Afterwards, m more XOR gates are needed to add these two results, which lead to one more T_X delay. To sum up, the space complexity of

TABLE 5
Space and time complexities of $S_2 \bmod f(x)$.

Operation		#AND	#XOR	Delay
(i)	$A_s + A_t$	-	$(n^2\ell + r^2 - m)/2$	T_X
	$B_s + B_t$	-	$(n^2\ell + r^2 - m)/2$	
(ii)	$D_{s,t}$ of ℓ bits	$\binom{n-r}{2}\ell^2$	$\binom{n-r}{2}(\ell-1)^2$	$\leq T_A + \lceil \log_2(\ell+3) \rceil T_X$
	$D_{s,t}$ of $\ell+1$ bits	$(\binom{n}{2} - \binom{n-r}{2})(\ell+1)^2$	$(\binom{n}{2} - \binom{n-r}{2})\ell^2$	
	$D_{s,t}$ additions	-	$\frac{n}{2} \cdot (m + r - 2 - 2\ell) + nr - \frac{r^2+r}{2}$ (even n) $\frac{n-1}{2} \cdot (m + r - 2 - \ell) + nr - \frac{r^2+r}{2}$ (odd n)	
(iii) (iv)	Case 2	-	$\frac{m(n-2)}{2} + (\frac{2n-t+1}{2})(r-2) + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert - \varepsilon$	$\lceil \log_2(\frac{n+4}{2} + \max\{\frac{n-t-1}{2}, \frac{t+1}{2}\}) \rceil$
	Case 3	-	$\frac{m(n-2)}{2} + n(r-2) + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert$	$\lceil \log_2(n+2) \rceil$
	Case 4 ($n = 3$)	-	$m + 2\ell + 2r - 4$	$\lceil \log_2 5 \rceil$
	Case 5	-	$\frac{m(n-3)}{2} + (\frac{2n-t-1}{2})(r+\ell-2) + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert - \varepsilon$	$\lceil \log_2(\frac{n+3}{2} + \max\{\frac{n-t-2}{2}, \frac{t+1}{2}\}) \rceil$
	Case 6	-	$\frac{m(n-3)}{2} + (n-1)(r+\ell-2) + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert$	$\lceil \log_2(n+1) \rceil$ (T_X)
where $\Delta_i = (n-2i+1)\ell - k$ if n is even or $(n-2i)\ell - k$ if n is odd, $\varepsilon = k - t\ell$, $t \geq 1$ is an odd integer that satisfy $t\ell \leq k$, $(t+2)\ell > k$.				

our proposed architecture is:

AND: $\frac{m^2}{2} + \frac{m\ell}{2} + (m + n + \frac{\ell+1}{2})r - (\ell+2)r^2$,

XOR: $\frac{m^2}{2} + (2n + \frac{\ell}{2} + r - 2)m + \frac{n^2+3rn+\ell r}{2} + \sum_{i=1}^{\ell} W(i)$
$+ \sum_{i=0}^{n-1} \lvert\theta_i\rvert + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert - \ell r^2 - \ell - \frac{7n+r^2}{2}$, ($n$ even),

or

$\frac{m^2}{2} + (2n + r + \frac{\ell-3}{2})m + \frac{n^2+3rn+\ell r+6}{2} + \sum_{i=1}^{\ell} W(i)$
$+ \sum_{i=0}^{n-1} \lvert\theta_i\rvert + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert - \ell r^2 - \frac{r^2+6r+3\ell+7n}{2}$, ($n$ odd).

The time delay (T_D) is

$$T_D \leq T_A + (2 + \lceil \log_2(\ell+3) \rceil + \lceil \log_2(n+2) \rceil) T_X.$$

It is noteworthy that in Table 5, there are several cases for the number of required XOR gates. For simplicity, we only present the maximum of required XOR gates. According to these formulations, we directly know that no matter which n-term KA (i.e., the choice of n, ℓ, r) we choose, the corresponding hybrid multiplier requires at least $\frac{m^2}{2}$ AND gates as well as $\frac{m^2}{2}$ XOR gates. Thus, it is the lower bound of the space complexity that our proposal can achieve. In fact, since the parameters n, ℓ, r and k all influence the space and time complexity, we can only obtain a certain optimal result under some preconditions. For example, if we consider minimizing the number of required AND gates only, ℓ should be equal to one. But in this case, we have $n = m$. The number of required XOR gates will be greater than $\frac{5m^2}{2}$.

In particular, since $m = n\ell + r$ (where $0 \leq r < n, 0 \leq r < \ell$, and r is less than $\sqrt{m}$), it is reasonable to stipulate that r is a small integer. Thus, the space complexity of our proposal depends on the selection of n, ℓ, k. Note that $\sum_{i=0}^{\ell} W(i)$ can be roughly written as $\frac{\ell}{2} \log_2 \ell$ [17]. If we ignore these small parts of above complexities formulae, the space complexity of our proposal is determined by some quadratic subexpressions.

4.3.1 Influence of parameter k

Although the irreducible trinomial $x^m + x^k + 1$ is usually given in advance, its term order k does have a significant impact on the space and time complexity. As we presented in Section 3.2, the time delay of adding these vectors $\mathbf{P}_i, \mathbf{P}'_i$ and $p_1(x)h(x)x^{-k}$ in parallel is $\lceil \log_2(n+1+\max\{t, n-t\}) \rceil$, where t satisfies $(t-1)\ell + \epsilon_{t-1} \leq k < t\ell + \epsilon_t$. It is obvious that when t approaches $n/2$, we obtain the minimal time delay. We then directly know that k is close to $(n/2) \cdot \ell \approx m/2$. Meanwhile, from Table 5, the computations of step (iii) and (iv) in this case also have lower gates delay.

Also notice that, in the space complexity formulae related to #XOR, the values of $\sum_{i=0}^{n-1} \lvert\theta_i\rvert$ and $\sum_{i=1}^{\lambda} \lvert\Delta_i\rvert$ ($\lambda = n/2$ for even n and $\lambda = (n-1)/2$ for odd n) are determined by k. In fact,

$$\sum_{i=0}^{n-1} \lvert\theta_i\rvert = \sum_{i=0}^{n-1} \lvert i\ell + \epsilon_i - k\rvert = tk + \sum_{i=t}^{n-1} (i\ell + \epsilon_i) - \sum_{i=0}^{t-1} (i\ell + \epsilon_i) - (n-t)k,$$

and

$$\sum_{i=1}^{\lambda} \lvert\Delta_i\rvert = \sum_{i=1}^{\lambda} \lvert(2\lambda - 2i + 1)\ell - k\rvert = \frac{(t'+1)k}{2} - \sum_{i=\frac{2\lambda-t'+1}{2}}^{\lambda} (2\lambda - 2i + 1)\ell + \sum_{i=1}^{\frac{2\lambda-t'-1}{2}} (2\lambda - 2i + 1)\ell - (\lambda - \frac{t'+1}{2})k,$$

where t satisfies $(t-1)\ell + \epsilon_{t-1} \leq k < t\ell + \epsilon_t$ and t' is an odd integer satisfying $t'\ell \leq k < (t'+2)\ell$. Please note that $t - 1$ is not always equal to t'.

In order to inspect the variation tendency of above expressions with respect to t and t', we expand these expressions by omitting the small parameter ϵ_i and recognize them as two functions:

$$f_1(t) = (2t - n)k + (-t^2 + t + \frac{n^2 - n}{2})\ell$$
$$f_2(t') = (t' + 1 - \lambda)k + \frac{(-t'^2 - 2t' + 2\lambda^2 - 1)\ell}{2}$$

Obviously, the bigger of the parameters t and t' are, the smaller of two functions become. That is to say, bigger k can lead to a lower space complexity. As a result, the trinomial $x^m + x^k + 1$ with k approaching to $\frac{m}{2}$ is more suitable to develop efficient hybrid Karatsuba multiplier. In fact, the authors of [21] already show that $x^m + x^{m/2} + 1$ combined with 2-term KA can develop a high efficient hybrid multiplier, which conform to this assertion.

4.3.2 Optimal selection of n, ℓ

From previous description, we know that k highly influences the values of $\sum_{i=0}^{n-1} \lvert\theta_i\rvert$ and $\sum_{i=1}^{\lambda} \lvert\Delta_i\rvert$. If k is fixed, the parameters n, ℓ can determine the space complexity of our proposal, so that we can obtain the optimal n and ℓ. Remember that r is usually chosen as a

small number. Its influence about the overall complexity is small. Thus, we do not consider it for simplicity.

If $k = 1$, then $t = 1, t' = -1$, $\sum_{i=0}^{n-1} |\theta_i|$ and $\sum_{i=1}^{\lambda} |\Delta_i|$ reach their maximum value, i.e.,

$$\max \sum_{i=0}^{n-1} |\theta_i| = \frac{n(n-1)\ell}{2} + \frac{r(r-1)}{2} - (n-2),$$
$$\max \sum_{i=1}^{\lambda} |\Delta_i| = \lambda^2\ell - \lambda, (\lambda = \frac{n}{2} \text{ or } \frac{n-1}{2}).$$

The magnitude of above subexpressions are both $O(n^2\ell)$. Without loss of generality, we consider the optimal n, ℓ under such a condition. In order to minimize both number of AND and XOR gates, we combine the two formulations with respect to #AND and #XOR, omit the small subexpressions, and define a function pertaining to overall logic gates:

$$M(n, \ell) = m^2 + (\frac{11n}{4} + \ell)m$$

where $\ell \approx \frac{m}{n}$. Obviously, if $11n = 4\ell$, $M(n, \ell)$ achieves its lower bound, which indicates the best asymptotic space complexity of our proposal. Now, the space complexity is

$$\#\text{ AND} = \frac{m^2}{2} + O\left(\frac{\sqrt{11}m^{3/2}}{4}\right)$$
$$\#\text{ XOR} = \frac{m^2}{2} + O\left(\frac{\sqrt{11}m^{3/2}}{2}\right)$$

Therefore, the optimal n, ℓ vary according to k. When k approaches to $m/2$, we can obtain other optimal n, ℓ, that result in even better space and time complexities. In practical applications, we note that for any given m, we cannot always select the exact optimal n, ℓ. But the flexible decomposition $m = n\ell + r$ can provide more alternative parameter choices. One can easily check all these possible decompositions and find the optimal parameters.

Table 6 provides a comparison of different bit-parallel multipliers for irreducible trinomials. All these multipliers are using PB representations, except particular description. It is clear that the space complexity of our scheme roughly matches the proposals of [18] and [32], and is lower than other previous architectures (quadratic or hybrid). The best of our results only costs about $m^2 + O(3\sqrt{11}m^{3/2}/4)$ logic gates (the total number of AND and XOR gates). In contrast, the time complexity of the proposed multiplier is slightly higher than the fastest result utilizing the classic Karatsuba algorithm. Also, strictly speaking, our scheme requires a little more XOR gates compared with [18] and [32]. The main reason is, we do not combine the polynomial multiplication and modular reduction together as the Mastrovito approach did, in order to obtain a relatively simpler architecture, which requires slightly more XOR gates and leads to more gate delays.

To illustrate the potential application of our proposal, we consider several irreducible trinomials already presented in [32], including the trinomials recommend by NIST [20]. Table 7 summarized the space and time complexity of our proposal and other schemes. Without loss of generality, we only compare our scheme with [18] and [32], which are also Karatsuba-based hybrid multipliers. One can check that, by choosing more flexible parameters n, ℓ, r, our proposal can achieve even better space and time trade-off for some finite fields. For instance, our proposal requires fewer logic gates (the plus of AND and XOR gates) compared with [18] and [32], for the fields $GF(2^{409}), GF(2^{431}), GF(2^{439}), GF(2^{447})$. For the fields $GF(2^{415})$ and $GF(2^{423})$, our scheme costs more logic gates than [32] but save one T_X delay. Meanwhile, the space complexity of our scheme is still less than [18]. For the field $GF(2^{233})$, our proposal costs fewer AND gates, but costs more XOR gates. So the total number of logic gates is greater than [32]. From the viewpoint of space complexity, [32] is preferred.

In the following section, we investigate possible speedup strategy for our scheme under a special class of trinomials.

5 Time complexity for Special Trinomials $x^{n\ell} + x^{t\ell} + 1, t > 1$

As shown in previous section, the time delay of our proposal is less than $T_A + (2 + \lceil \log_2(\ell + 3) \rceil + \lceil \log_2(n + 2) \rceil)T_X$. But we note that

$$\lceil \log_2(\ell + 3) \rceil + \lceil \log_2(n + 2) \rceil \le 1 + \lceil \log_2(\ell + 3)(n + 2) \rceil = 1 + \lceil \log_2(m + 2\ell + 3n + 6) \rceil.$$

The upper bound of the delay of our architecture is bigger than $T_A + (3 + \lceil \log_2 m \rceil)T_X$, which at most matches the classic hybrid Karatsuba multiplier [11]. In order to obtain a lower time complexity, we want to apply a speedup strategy to our architecture, which was proposed in [30]. However, the precondition to apply such a speedup strategy is that delay of $S_1 x^{-2k} \bmod f(x)$ is lower than that of $S_2 x^{-2k} \bmod f(x)$ by at least one T_X. If these delays are equal, no speedup can achieve. In [30], the authors utilized a special type of trinomial, i.e., $f(x) = x^{nk} + x^k + 1$. In this scenario, the corresponding $S_1 x^{-2k} \bmod f(x)$ can be performed by a simple matrix-vector that requires a lower time complexity than ordinary cases. Besides above special type of trinomial, in this section, we show that another type of trinomial, i.e., $x^m + x^k + 1, m = n\ell, k = t\ell, t > 1$ can also provide a better space and time complexity trade-off, and apply speedup strategy under a certain condition.

In this case, we have $r = 0$. The computations of p_0 and p_1 are the same as those presented in (3). Nevertheless, the subexpressions in (6) and (7) now have some common terms, which can save certain logic gates. More explicitly, since $k = t\ell$, we have

$$\begin{aligned} E'(x)x^{i\ell-2t\ell} \bmod f(x) = & \underline{\sum_{j=(t-i)\ell}^{n\ell-1} e'_j x^{j+(i-2t)\ell}} \\ & + \sum_{j=0}^{(t-i)\ell-1} e'_j x^{j+(i+n-2t)\ell} + \underline{\sum_{j=0}^{(t-i)\ell-1} e'_j x^{j+(i-t)\ell}}, \\ & \cdots\cdots\cdots\cdots\cdots\cdots \end{aligned} \tag{18}$$

for $i = 0, 1, \cdots, t-1$, and

$$\begin{aligned} & \cdots\cdots\cdots\cdots\cdots\cdots \\ E'(x)x^{i\ell-2t\ell} \bmod f(x) = & \sum_{j=0}^{(n-i+t)\ell-1} e'_j x^{j+(i-2t)\ell} \\ & + \sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-2t-n)\ell} + \underline{\sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-n-t)\ell}}, \end{aligned} \tag{19}$$

for $i = t, t+1, \cdots, n-1$. Notice that if $i = t$, the subexpression $\sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-n-t)\ell} = \sum_{j=n\ell}^{n\ell-1} e'_j x^{j-n\ell}$

TABLE 6
Comparison of Some Bit-Parallel Multipliers for Irreducible Trinomials $x^m + x^k + 1, m \geq 2k$.

Multiplier	# AND	# XOR	Time delay
Montgomery [27], school-book [26] Mastrovito [22] [23] [24]	m^2	$m^2 - 1$	$T_A + (2 + \lceil \log_2 m \rceil)T_X$
Mastrovito [25]	m^2	$m^2 - 1$	$T_A + (\lceil \log_2(2m + 2k - 3) \rceil)T_X$
SPB Mastrovito [13] Montgomery [14]	m^2	$m^2 - 1$	$T_A + \lceil \log_2(2m - k - 1) \rceil T_X$
KA [11]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + 4m + k - \frac{23}{4}$ (m odd)	$T_A + (3 + \lceil \log_2(m-1) \rceil)T_X$
	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + \frac{5m}{2} + k - 4$ (m even)	
Modified KA [15]	$\frac{m^2}{2} + (m-k)^2$	$\frac{m^2}{2} + (m-k)^2 + 2k$	$T_A + (2 + \lceil \log_2(m-1) \rceil)T_X$
Modified KA [10]	$m^2 - k^2$	$m^2 + k - k^2 - 1 (1 < k < \frac{m}{3})$	$\leq T_A + (2 + \lceil \log_2 m \rceil)T_X$
		$m^2 + 4k - k^2 - m - 1 (\frac{m}{3} \leq k < \frac{m-1}{2})$	
		$m^2 + 2k - k^2 (k = \frac{m-1}{2})$	
Montgomery squaring [17]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + O(m \log_2 m)$ (m odd)	$\leq T_A + (3 + \lceil \log_2 m \rceil)T_X$
	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + O(m \log_2 m)$ (m even)	$T_A + (2 + \lceil \log_2 m \rceil)T_X$
Chinese Remainder Theorem [28]	Δ	$\Delta + 3k - m$ (Type-A)	$T_A + \lceil \log_2(\Theta) \rceil T_X$
	Δ	$\Delta + 2k - m + kW(k)$ (Type-B)	$T_A + \lceil \log_2(3m - 3k - 1) \rceil T_X$
SPB Mastrovito-KA [18]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + \frac{m}{2} + O(m \log_2 m)$ (m odd)	$T_A + (1 + \lceil \log_2(2m-k-1) \rceil)T_X$
	$\frac{3m^2}{4}$	$\frac{3m^2}{4} - \frac{m}{2} + O(m \log_2 m)$ (m even)	
SPB Mastrovito n-term KA [30] Trinomial of $m = nk$	$\frac{m^2}{2} + \frac{mk}{2}$	$\frac{m^2}{2} + \frac{mk}{2} + \frac{5mn}{4} + O(m \log_2 k)$	$T_A + (\lceil \log_2 k \rceil + \lceil \log_2 3n \rceil)T_X$
SPB Mastrovito n-term KA [32] $m = n\ell, n\ell + 1$	$\frac{m^2}{2} + O(\frac{m\ell}{2})$	$\frac{m^2}{2} + \frac{m\ell}{2} + \frac{5mn}{4} + O(m \log_2 \ell)$	$\leq T_A + (2 + \lceil \log_2 \ell \rceil + \lceil \log_2 n \rceil)T_X$
This paper $m = n\ell + r$	$\frac{m^2}{2} + O(\frac{m\ell}{2})$	$\frac{m^2}{2} + \frac{m\ell}{2} + \frac{11mn}{4} + O(m)$	$\leq T_A + (2 + \lceil \log_2(\ell + 3) \rceil + \lceil \log_2(n + 2) \rceil)T_X$
where $\Delta = m^2 + \frac{(m-k)(m-1-3k)}{2}$ ($\frac{m-1}{3} \leq k < \frac{m}{2}, 2^{v-1} < k \leq 2^v$), $\Theta = \max(3m-3k-1, 2m-2k+2^v)$			

TABLE 7
Space and time complexities comparison for some trinomials.

m, k	Multiplier	n, ℓ, r	#AND	#XOR	Delay	#AND+#XOR	Description
$233, 74$	[18]	-	40833	44036	$T_A + 10T_X$	84869	-
	[32](k=159)	(8, 29, 1)	30741	33983	$T_A + 11T_X$	64724	-
	Proposal	(21, 11, 2)	28894	38090	$T_A + 11T_X$	66984	fewer #AND
$409, 87$	[18]	-	125665	131822	$T_A + 11T_X$	257487	-
	[32](k=322)	(51, 8, 1)	85681	117268	$T_A + 11T_X$	202949	-
	Proposal	(15, 27, 4)	90450	109812	$T_A + 11T_X$	200262	fewer gates
$415, 163$	[18]	-	129376	135150	$T_A + 11T_X$	264526	-
	[32] (k=252)	(18, 23, 1)	91288	103175	$T_A + 12T_X$	194463	-
	Proposal	(18, 23, 1)	91305	110510	$T_A + 11T_X$	201815	less delay
$423, 25$	[18]	-	134408	141543	$T_A + 11T_X$	275951	-
	[32]	(9, 47, 0)	99405	104212	$T_A + 12T_X$	203617	-
	Proposal	(15, 28, 3)	96474	117918	$T_A + 11T_X$	214392	less delay
$431, 200$	[18]	-	139536	145752	$T_A + 11T_X$	296542	-
	[32] (k=231)	(215, 2, 1)	93741	224114	$T_A + 11T_X$	317855	-
	Proposal	(39, 11, 2)	96151	123857	$T_A + 11T_X$	220008	fewer gates
$439, 171$	[18]	-	144760	151782	$T_A + 11T_X$	296542	-
	[32](k=294)	(6, 73, 1)	112786	117249	$T_A + 12T_X$	230035	-
	Proposal	(23, 19, 2)	101391	123086	$T_A + 12T_X$	224477	fewer gates
$447, 83$	[18]	-	150080	157817	$T_A + 11T_X$	307897	-
	[32]	(3, 149, 0)	133206	134530	$T_A + 12T_X$	267736	-
	Proposal	(26, 17, 5)	105639	130198	$T_A + 12T_X$	235837	fewer gates

does not exist. To find the common terms among above expressions, we let $i \in \{0, 1, \cdots, t-1\}$ and $i' \in \{t, t+1, \cdots, n-1\}$. Also note that $m \geq 2k \Rightarrow n \geq 2t \Rightarrow n-t \geq t$. The former group has fewer items than the latter. When comparing the subexpressions in (18) and (19), we found that if $i'-i=t$, subexpressions $\sum_{j=0}^{(t-i)\ell-1} e'_j x^{j+(i-t)\ell}$ have common terms with $\sum_{j=0}^{(n-i'+t)\ell-1} e'_j x^{j+(i'-2t)\ell}$. In this case,

$$\sum_{j=0}^{(t-i)\ell-1} e'_j x^{j+(i-t)\ell} + \sum_{j=0}^{(n-i'+t)\ell-1} e'_j x^{j+(i'-2t)\ell} = \sum_{j=(2t-i')\ell}^{(n-i'+t)\ell-1} e'_j x^{j+(i'-2t)\ell}.$$

for $i'=n-t, n-t+1, \cdots, n-1$. Similarly, if $i'-i=n-t$, the subexpressions $\sum_{j=(t-i)\ell}^{n\ell-1} e'_j x^{j+(i-2t)\ell}$ have common terms with $\sum_{j=(n-i'+t)\ell}^{n\ell-1} e'_j x^{j+(i'-n-t)\ell}$ as well:

$$\sum_{j=(t-i)\ell}^{n\ell-1} e'_j x^{j+(i-2t)\ell} + \sum_{j=(n-i'+t)\ell}^{n\ell-1} e'_j x^{j+(i'-n-t)\ell} = \sum_{j=(t-i)\ell}^{(2t-i)\ell-1} e'_j x^{j+(i-2t)\ell}.$$

for $i = 0, 1, \cdots, t-1$. Particularly, we add different styles of underlines to these subexpressions in order to indicate the overlapped parts. One can check that all the dotted underlined subexpressions in (18) can be eliminated by offsetting related expressions in (19), but only t solid underlined subexpressions in (19) can be eliminated. After combining the overlapped parts between (18) and (19), the rest of subexpressions can be rewritten as $n+n-t=2n-t$ coordinate vectors $\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_{2n-2t-1}$, where

$$\mathbf{p}_i = \sum_{j=(t-i)\ell}^{(2t-i)\ell-1} e'_j x^{j+(i-2t)\ell} + \sum_{j=0}^{(t-i)\ell-1} e'_j x^{j+(i+n-2t)\ell},$$

for $i = 0, 1, \cdots, t-1$,

$$\mathbf{p}_i = \sum_{j=0}^{(n-i+t)\ell-1} e'_j x^{j+(i-2t)\ell} + \sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-2t-n)\ell},$$

for $i = t, \cdots, n-t-1$,

$$\mathbf{p}_i = \sum_{j=(2t-i)\ell}^{(n-i+t)\ell-1} e'_j x^{j+(i-2t)\ell} + \sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-2t-n)\ell},$$

$i = n-t, \cdots, n-1$, and

$$\mathbf{p}_i = \sum_{j=(2n-i)\ell}^{n\ell-1} e'_j x^{j+(i-2n)\ell},$$

for $i = n, \cdots, 2n-2t-1$. Specifically, one can easily check that $\mathbf{p}_i, (i = 0, 1, \cdots, t-1)$ have no overlap with $\mathbf{p}_n, \cdots, \mathbf{p}_{2n-2t-1}$. Please notice that $n \geq 2t \Rightarrow 2n-2t-1 \geq 2t-1 \geq t-1$. Thus, some of these vectors as above can be combined and rewritten as $2n-3t$ independent vectors

$$\mathbf{p}_0 + \mathbf{p}_n, \mathbf{p}_1 + \mathbf{p}_{n+1}, \cdots, \mathbf{p}_{t-1} + \mathbf{p}_{n+t-1},$$
$$\mathbf{p}_t, \cdots, \mathbf{p}_{n-1}, \mathbf{p}_{n+t}, \cdots, \mathbf{p}_{2n-2t-1},$$

without any logic gates. As a result, the addition between (18) and (19) can be implemented by adding $2n-3t$ subexpressions in parallel. Also note that $p_1(x)h(x)x^{-k}$ needs to be added. Plus the delay of the computation of p_1, p_0+p_1 presented in Table 3, the computation of $S_1 x^{-2k} \bmod f(x)$ here requires $T_A + (\lceil \log_2 \ell \rceil + \lceil \log_2(2n-3t+1) \rceil) T_X$ delays.

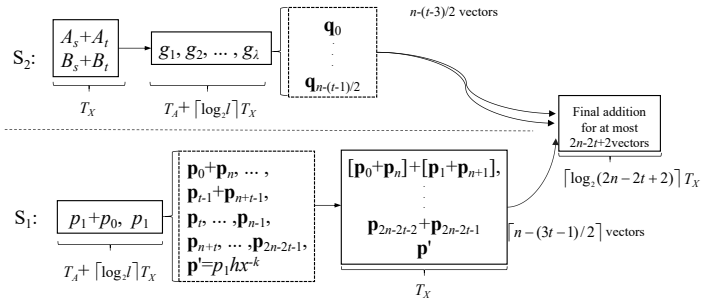

Fig. 3. Speedup Strategy for $x^{n\ell} + x^{t\ell} + 1$.

Conversely, when we consider the delay of $S_2 x^{-2k} \bmod f(x)$ here, it is easy to check that the computations of Step (i)-(iv) are the same as that shown in Section 4.2. Please note that $m = n\ell, k = t\ell, n \geq 2t, t > 1$. The magnitude relations of n, ℓ, k only satisfy Case 2 and 5. Then, one can check that the time delay of $S_2 x^{-2k} \bmod f(x)$ is $T_A + (1 + \lceil \log_2 \ell \rceil + \lceil \log_2(n - \frac{t-3}{2}) \rceil) T_X$. Clearly, $1 + \lceil \log_2(n - \frac{t-3}{2}) \rceil \geq \lceil \log_2(2n-3t+1) \rceil$. Therefore, the implementation of $S_1 x^{-2k} \bmod f(x)$ is faster than $S_2 x^{-2k} \bmod f(x)$. But it is especially interesting if

$$1 + \lceil \log_2(n - \frac{t+3}{2}) \rceil > \lceil \log_2(2n-3t+1) \rceil$$

We have checked all the n in the range $[3, 100]$ and found that if $n \geq 6, n \neq 7$, there exists at least one t to make the above inequation hold. In this case, the delay of $S_1 x^{-2k} \bmod f(x)$ is one T_X lower that of $S_2 x^{-2k} \bmod f(x)$, which can apply the same speedup strategy presented in [30]. The key idea of such a strategy is adding the intermediate values in advance during the computation process of S_1 and S_2. More explicitly, let $\mathbf{q}_0, \mathbf{q}_1, \cdots, \mathbf{q}_{n-(t-1)/2}$ denote coordinate vectors corresponding to the subexpressions in (11), (12), (16), (17). Instead of adding $S_1 x^{-2k} \bmod f(x)$ and $S_2 x^{-2k} \bmod f(x)$ to obtain the final result, we can add $\mathbf{q}_i$ and $\mathbf{p}_i$ directly.

From Figure 3, the elements in the dot line box do not cost any logic gates as all the vectors are obtained by reorganizing the intermediate values of former steps. After applying speedup strategy, the logic gates delay for the whole multiplier is

$$T_A + (2 + \lceil \log_2 \ell \rceil + \lceil \log_2(n-t+1) \rceil) T_X$$

which matches the results of [32]. Also, since some subexpresssions in (18) and (19) can offset, certain numbers of XOR gates can be saved. But this number is small, which has little impact on the overall space complexity. The study of optimal n, ℓ can follow the same line, as in Section 4.3.2.

6 Conclusion

In this paper, we extend the application of an n-term Karatsuba algorithm for general trinomials $x^m + x^k + 1$, by decomposing m into $m = n\ell + r$. Under such a decompositifon, the m-term polynomial multiplication is reorganized in order to apply n-term KA. Then, a new type of hybrid Karatsuba $GF(2^m)$ multiplier architecture is

proposed. We give the explicit space and time complexity formulations and evaluate the upper and lower bounds. The optimal choices of the KA parameters as well as the irreducible trinomial are investigated. Consequently, the space complexity of our proposal can achieve to $m^2/2 + O(\sqrt{11}m^{3/2}/2)$, which matches the best result of current hybrid multipliers. Meanwhile, its time complexity is slightly higher than the counterparts. In addition, we also investigated possible speedup strategy for special trinomials. A new type of trinomials is considered to simplify the modular reduction and further speed up related multipliers. The corresponding time complexity now matches the results of [30], [32]. To find more special types of polynomial that can lead to better space and time complexity trade-off would be the future work.

REFERENCES

[1] R. Lidl *et al. Finite Fields*, Cambridge University Press, New York, NY, USA, 1996.

[2] A. Karatsuba *et al.* "Multiplication of Multidigit Numbers on Automata," *Soviet Physics-Doklady (English translation)*, vol. 7, no. 7, pp. 595–596, 1963.

[3] D. Knuth. *The Art of Computer Programming*, Volume 2. Third Edition, Addison-Wesley, 1997.

[4] J. Von Zur Gathen *et al. Modern Computer Algebra (2 ed.)*. Cambridge University Press, New York, NY, USA.

[5] P.L. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," *IEEE Trans. on Computers*, vol. 54, no. 3, pp. 362-369, 2005.

[6] H. Fan *et al.* "Overlap-free Karatsuba-Ofman polynomial multiplication algorithms," *Information Security*, vol. 4, no. 1, pp. 8–14, 2010.

[7] A. Weimerskirch *et al.* "Generalizations of the Karatsuba Algorithm for Efficient Implementations," *Cryptology ePrint Archive, Report 2006/224,* http://eprint.iacr.org/

[8] H. Fan *et al.* "Obtaining more Karatsuba-like formulae over the binary field," *IET Information Security*, vol. 6, no. 1, pp. 14-19, 2012.

[9] K. Chang *et al.* "Low complexity bit-parallel multiplier for $GF(2^m)$ defined by all-one polynomials using redundant representation," *IEEE Trans. Comput.*, vol. 54, no. 12, pp. 1628–1630, 2005.

[10] Y. Cho *et al.* "New bit parallel multiplier with low space complexity for all irreducible trinomials over $GF(2^n)$," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 10, pp. 1903–1908, Oct 2012.

[11] M. Elia *et al.* "Low complexity bit-parallel multipliers for $GF(2^m)$ with generator polynomial $x^m + x^k + 1$," *Electronic Letters*, vol. 35, no. 7, pp. 551–552, 1999.

[12] H. Fan *et al.* "Fast bit-parallel $GF(2^n)$ multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 485–490, 2005.

[13] H. Fan *et al.* "Fast bit parallel-shifted polynomial basis multipliers in $GF(2^n)$," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, no. 12, pp. 2606–2615, 2006.

[14] A. Hariri *et al.* "Bit-serial and bit-parallel montgomery multiplication and squaring over $GF(2^m)$," *IEEE Trans. on Computers*, vol. 58, no. 10, pp. 1332–1345, 2009.

[15] Y. Li *et al.* "Speedup of bit-parallel karatsuba multiplier in $GF(2^m)$ generated by trinomials," *Information Processing Letters*, vol. 111, no. 8, pp. 390–394, 2011.

[16] H. Fan *et al.* "A survey of some recent bit-parallel multipliers," *Finite Fields and Their Applications*, vol. 32, pp. 5–43, 2015.

[17] Y. Li *et al.* "New bit-parallel Montgomery multiplier for trinomials using squaring operation," *Integration, the VLSI Journal*, vol. 52, pp.142–155, January 2016.

[18] Y. Li *et al.* "Mastrovito Form of Non-recursive Karatsuba Multiplier for All Trinomials," *IEEE Trans. Comput.*, vol. 66, no.9, pp.1573–1584, Sept. 2017.

[19] C. Negre. "Efficient parallel multiplier in shifted polynomial basis," *J. Syst. Archit.*, vol. 53, no. 2-3, pp. 109–116, 2007.

[20] Recommended elliptic curves for Federal Government use, http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, July, 1999.

[21] H. Shen *et al.* "Low complexity bit parallel multiplier for $GF(2^m)$ generated by equally-spaced trinomials," *Inf. Process. Lett.*, vol. 107, no. 6, pp. 211–215, 2008.

[22] B. Sunar *et al.* "Mastrovito multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 48,no. 5, pp. 522–527, 1999.

[23] A. Halbutogullari *et al.* "Mastrovito multiplier for general irreducible polynomials," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 503–518, May 2000.

[24] T. Zhang *et al.* "Systematic design of original and modified mastrovito multipliers for general irreducible polynomials," *IEEE Trans. Comput.*, vol. 50, no. 7, pp. 734–749, 2001.

[25] N. Petra, *et al.* "A novel architecture for galois fields $GF(2^m)$ multipliers based on mastrovito scheme," *IEEE Trans. Computers*, vol. 56, no. 11, pp. 1470–1483, November 2007.

[26] H. Wu. "Bit-parallel finite field multiplier and squarer using polynomial basis," *IEEE Trans. Comput.*, vol. 51, no. 7, pp. 750–758, 2002.

[27] H. Wu. "Montgomery multiplier and squarer for a class of finite fields," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 521–529, 2002.

[28] H. Fan, "A Chinese Remainder Theorem Approach to Bit-Parallel $GF(2^n)$ Polynomial Basis Multipliers for Irreducible Trinomials", *IEEE Trans. Comput.*, vol. 65, no. 2, pp. 343–352, February 2016.

[29] X. Xie, *et al.* "Novel bit-parallel multiplierfor $GF(2^m)$ defined by all-one polynomial using generalized Karatsuba algorithm", *Information Processing Letters*, vol. 114, no. 3, pp.140–146, 2014.

[30] Y. Li, *et al.* "N-Term Karatsuba Algorithm and Its Application to Multiplier Designs for Special Trinomials," *IEEE Access*, vol. 6, pp.43056–43069, Jul. 2018.

[31] Y. Li, *et al.* "Efficient Nonrecursive Bit-Parallel Karatsuba Multiplier for a Special Class of Trinomials," *VLSI Design*, vol. 2018, Article ID 9269157, 7 pages, 2018.

[32] S. Park, *et al.* "Low Space Complexity $GF(2^m)$ Multiplier for Trinomials Using n-Term Karatsuba Algorithm," in IEEE Access, vol. 7, pp. 27047-27064, 2019.

[33] H. Fan *et al.* "Relationship between $GF(2^m)$ Montgomery and Shifted Polynomial Basis Multiplication Algorithms," *IEEE Trans. on Computers*, vol. 55, no. 9, pp. 1202-1206, 2006.

APPENDIX A
A SMALL EXAMPLES OF n-TERM KARATSUBA MULTIPLIER

In the following, we give the computation details presented in Table 1.

A.1 Computation of I of Table 1

I.(i): Instead of computing $A_2B_2x^4 + A_1B_1x^2 + A_0B_0$ directly, we partition this formula into two parts, i.e., $p_1x^7 + p_0$:

$$A_2B_2x^4 + A_1B_1x^2 + A_0B_0 =$$
$$\begin{matrix} \\ p_0: \\ \\ \\ \\ \\ \\ \\ p_1: \end{matrix} \left[\begin{array}{ccc:ccc:ccc} a_0 & 0 & & 0 & 0 & & 0 & 0 & 0 \\ a_1 & a_0 & & 0 & 0 & & 0 & 0 & 0 \\ 0 & a_1 & & a_2 & 0 & & 0 & 0 & 0 \\ 0 & 0 & & a_3 & a_2 & & 0 & 0 & 0 \\ 0 & 0 & & 0 & a_3 & & a_4 & 0 & 0 \\ 0 & 0 & & 0 & 0 & & a_5 & a_4 & 0 \\ 0 & 0 & & 0 & 0 & & a_6 & a_5 & a_4 \\ \hdashline 0 & 0 & & 0 & 0 & & 0 & a_6 & a_5 \\ 0 & 0 & & 0 & 0 & & 0 & 0 & a_6 \end{array}\right] \times \left[\begin{array}{c} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{array}\right].$$

Then, $A_2B_2x^4+A_1B_1x^2+A_0B_0 \bmod x^7+x^3+1 = p_1x^7+p_0 \bmod x^7+x^3+1 = p_1x^3 + (p_0 + p_1)$. Obviously, the

computation of $p_0 + p_1$ is equivalent to the matrix-vector multiplication as follows:

$$p_0 + p_1 = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{bmatrix} = \left[\begin{array}{cc:cc:ccc} a_0 & 0 & 0 & 0 & 0 & a_6 & a_5 \\ a_1 & a_0 & 0 & 0 & 0 & 0 & a_6 \\ 0 & a_1 & a_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_3 & a_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_3 & a_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_5 & a_4 & 0 \\ 0 & 0 & 0 & 0 & a_6 & a_5 & a_4 \end{array}\right] \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}.$$

Meanwhile, the computation of p_1 is equivalent to

$$p_1 = \begin{bmatrix} e_7 \\ e_8 \end{bmatrix} = \begin{bmatrix} a_6 & a_5 \\ 0 & a_6 \end{bmatrix} \times \begin{bmatrix} b_5 \\ b_6 \end{bmatrix}.$$

We can check that $p_0 + p_1$ includes all the entries of p_1, and no AND and XOR gates are required using *sub-expression sharing* trick.

I.(ii): The modular multiplication of p_0+p_1, p_1x^3 with $(x^4+x^2+1)\cdot x^{-6}$ can be recognized shifts of $\mathbf{p}_0 + \mathbf{p}_1, \mathbf{p}_1$ and additions among them. We immediately have

$$(p_0 + p_1)\cdot(x^4+x^2+1)\cdot x^{-6} = \begin{bmatrix} e_6 \\ e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_0 \end{bmatrix} + \begin{bmatrix} e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_0 \\ e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ e_6 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ e_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

and

$$(p_1x^3)\cdot(x^4+x^2+1)\cdot x^{-6} = \begin{bmatrix} e_7 \\ e_8 \\ e_7 \\ e_8 \\ e_7 \\ e_8 \\ 0 \end{bmatrix}.$$

Therefore, the computation of I.(ii) can be implemented by the additions of seven vectors.

A.2 Computation of II of Table 1

I.(ii): Let $\sum_{i=0}^{2} u_i^{(s,t)}x^i$ denote the results of $A_2+A_1, A_2+A_0, A_1+A_0$ and $\sum_{i=0}^{2} v_i^{(s,t)}x^i$ denote the results of $B_2+B_1, B_2+B_0, B_1+B_0$. So,

$$\begin{array}{lll} u_0^{(2,1)} = a_4 + a_2 & u_0^{(2,0)} = a_4 + a_1 & u_0^{(1,0)} = a_2 + a_1 \\ u_1^{(2,1)} = a_5 + a_3, & u_1^{(2,0)} = a_5 + a_0, & u_1^{(1,0)} = a_3 + a_0, \\ u_2^{(2,1)} = a_6 & u_2^{(2,0)} = a_6 & u_2^{(1,0)} = 0 \end{array}$$

and

$$\begin{array}{lll} v_0^{(2,1)} = b_4 + b_2 & v_0^{(2,0)} = b_4 + b_1 & v_0^{(1,0)} = b_2 + b_1 \\ v_1^{(2,1)} = b_5 + b_3, & v_1^{(2,0)} = b_5 + b_0, & v_1^{(1,0)} = b_3 + b_0, \\ v_2^{(2,1)} = b_6 & v_2^{(2,0)} = b_6 & v_2^{(1,0)} = 0 \end{array}$$

All these operations can be implemented in parallel using XOR gates and finished in one T_X delay.

I.(ii): Analogous to the computation of Step I, $D_{2,1}x^4 + D_{2,0}x^2 + D_{1,0}$ can also be computed as a matrix-vector multiplication. We also split this expression into two parts and deal with their reduction, independently.

$$D_{2,1}x^4 + D_{2,0}x^2 + D_{1,0} = p_1^{(1)}x^7 + p_0^{(1)}$$

$$\left[\begin{array}{cc:ccc:ccc} u_0^{(1,0)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ u_1^{(1,0)} & u_0^{(1,0)} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u_1^{(1,0)} & u_0^{(2,0)} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_1^{(2,0)} & u_0^{(2,0)} & 0 & 0 & 0 & 0 \\ 0 & 0 & u_2^{(2,0)} & u_1^{(2,0)} & u_0^{(2,0)} & u_0^{(2,1)} & 0 & 0 \\ 0 & 0 & 0 & u_2^{(2,0)} & u_1^{(2,0)} & u_1^{(2,1)} & u_0^{(2,1)} & 0 \\ 0 & 0 & 0 & 0 & u_2^{(2,0)} & u_2^{(2,1)} & u_1^{(2,1)} & u_0^{(2,1)} \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & u_2^{(2,\overline{1})} & u_1^{(2,\overline{1})} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_2^{(2,1)} \end{array}\right] \times \left[\begin{array}{c} v_0^{(1,0)} \\ v_1^{(1,0)} \\ \hdashline v_0^{(2,\overline{0})} \\ v_1^{(2,0)} \\ v_2^{(2,0)} \\ \hdashline v_0^{(2,\overline{1})} \\ v_0^{(2,1)} \\ v_1^{(2,1)} \\ v_2^{(2,1)} \end{array}\right].$$

Then we have $(D_{2,1}x^6 + D_{2,0}x^4 + D_{1,0}x^2)\cdot x^{-6} \bmod x^7+x^3+1 = (p_1^{(1)}x^3 + p_0^{(1)} + p_1^{(1)})x^{-4} \bmod x^7+x^3+1)$. Such a expression can also be obtained using the additions of three vectors, i.e.,

$$\begin{aligned}&(p_1^{(1)}x^3 + p_0^{(1)} + p_1^{(1)})x^{-4} \bmod x^7+x^3+1 \\ &= \begin{bmatrix} h_1^{(1)} \\ h_2^{(1)} \\ h_3^{(1)} \\ h_4^{(1)} \\ h_5^{(1)} \\ h_6^{(1)} \\ h_0^{(1)} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ h_0^{(1)} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ h_7^{(1)} \\ h_8^{(1)} \\ 0 \\ 0 \\ h_7^{(1)} \\ h_8^{(1)} \end{bmatrix},\end{aligned}$$

where $h_i^{(1)}, i = 0, 1, \cdots, 8$ denote the coefficients of $D_{2,1}x^4 + D_{2,0}x^2 + D_{1,0}$.

In parallel implementation, it is easy to check that both of Step I and II cost $T_A + 6T_X$ delay, thus the whole circuit is $T_A + 6T_X$, where T_A and T_X denote the delay of AND and XOR gate, respectively. This result meets the delay assertions presented in section 3.2 and Proposition 3 of the main paper. Finally, another T_X delay is required to add the results of S_1x^{-6} and S_2x^{-6}.

APPENDIX B
PROOFS

B.1 Proof of Proposition 1

Proof For simplicity, we rewrite the formulae of A, B as follows:

$$A = A_{n-1}x^{\overline{n-1}} + A_{n-2}x^{\overline{n-2}} + \cdots + A_1x^{\overline{1}} + A_0x^{\overline{0}},$$
$$B = B_{n-1}x^{\overline{n-1}} + B_{n-2}x^{\overline{n-2}} + \cdots + B_1x^{\overline{1}} + B_0x^{\overline{0}},$$

where $\bar{i} = (\ell+1)i-n+r$ for $i = n-r+1, \cdots, n-1$ and $\bar{i} = \ell i$ for $i = 0, 1, \cdots, n-r$. The expansion of AB is

$$\begin{aligned} AB &= \sum_{i=0}^{n-1} A_i B_{n-1} x^{\bar{i}+\overline{n-1}} + \cdots + \sum_{i=0}^{n-1} A_i B_0 x^{\bar{i}+\bar{0}} \\ &= \sum_{i=0}^{n-1} A_i B_i x^{2\bar{i}} + \sum_{0\le i<j<n} (A_i B_j + A_j B_i) x^{\bar{i}+\bar{j}} \end{aligned} \tag{20}$$

Applying (1) in the paper, we know that $(A_iB_j + A_jB_i)x^{\bar{i}+\bar{j}} = ((A_i+A_j)(B_i+B_j)+A_iB_i+A_jB_j)x^{\bar{i}+\bar{j}}$. Plug these formulae into above expression, (20) can be rewritten as:

$$\begin{aligned} AB =& A_{n-1}B_{n-1}x^{\overline{n-1}}(x^{\overline{n-1}} + x^{\overline{n-2}} + \cdots + x^{\bar{1}} + 1) \\ &+ A_{n-2}B_{n-2}x^{\overline{n-2}}(x^{\overline{n-1}} + x^{\overline{n-2}} + \cdots + x^{\bar{1}} + 1) \\ &+ \cdots + A_1B_1x^{\bar{1}}(x^{\overline{n-1}} + x^{\overline{n-2}} + \cdots + x^{\bar{1}} + 1) \\ &+ A_0B_0x^{\bar{0}}(x^{\overline{n-1}} + x^{\overline{n-2}} + \cdots + x^{\bar{1}} + 1) \\ &+ \sum_{i=1}^{2n-3} \Big(\sum_{\substack{s+t=i, \\ n>s>t\ge 0}} D_{s,t} \Big) x^{\bar{s}+\bar{t}}. \end{aligned}$$

When we substitute the symbol $\bar{i}$ with the original degree, the conclusion is direct. □

B.2 Proof of Lemma 1

Proof The proof of this lemma mainly utilizes the reduction formulation (7). If the parameter $1 \le \Delta \le m-k-1$, we have

$$\begin{aligned} A(x)\cdot x^{\Delta} &= \sum_{i=0}^{m-1} a_i x^{i+\Delta-k} \\ &= \sum_{i=0}^{m-\Delta-1} a_i x^{i+\Delta-k} + \sum_{m-\Delta}^{m-1} a_i x^{i+\Delta-k} \\ &= \sum_{i=0}^{m-\Delta-1} a_i x^{i+\Delta-k} + \sum_{m-\Delta}^{m-1} (a_i x^{i+\Delta-m} + a_i x^{i+\Delta-m-k}) \\ &= \sum_{i=0}^{m-1} a_i x^{-k+(i+\Delta) \bmod m} + \sum_{i=m-\Delta}^{m-1} a_i x^{i+\Delta-m}. \end{aligned}$$

Similarly, if $-k \le \Delta < 0$, then $0 < -\Delta \le k$, we have

$$\begin{aligned} A(x)\cdot x^{\Delta} &= \sum_{i=0}^{m-1} a_i x^{i+\Delta-k} \\ &= \sum_{i=-\Delta}^{m-1} a_i x^{i+\Delta-k} + \sum_{i=0}^{-\Delta-1} a_i x^{i+\Delta-k} \\ &= \sum_{i=-\Delta}^{m-1} a_i x^{i+\Delta-k} + \sum_{i=0}^{-\Delta-1} (a_i x^{i+\Delta+m-k} + a_i x^{i+\Delta}) \\ &= \sum_{i=0}^{m-1} a_i x^{-k+(i+\Delta) \bmod m} + \sum_{i=0}^{-\Delta-1} a_i x^{i+\Delta}. \end{aligned}$$

We then directly conclude this lemma. □

B.3 Proof of Observation 1

Proof Apparently, the modular reductions of $g_1 x^{(2\lambda-1)\ell-2k}$, $g_2 x^{(2\lambda-3)\ell-2k}, \cdots, g_\lambda x^{\ell-2k}$ rely on their maximum and minimum term degrees.

Firstly, according to the explicit form of $g_1, g_2, \cdots, g_\lambda$, one can check that the degrees of the subexpressions $D_{s,t}\cdot x^{\delta_{s,t}}$ are in the range $[2\ell-2, 2\ell+2r-3]$, as $\deg(D_{s,t}) = 2\ell-2$ (for $0 \le t < s < n-r$) or 2ℓ (for $0 < t < s, s \ge n-r$) and $\max \delta_{s,t} = (n-1)+(n-2)-2(n-r) = 2r-3$. Then, it is easy to see that the term degrees of $g_1 x^{(2\lambda-1)\ell-2k}, \cdots, g_\lambda x^{\ell-2k}$ are all in the range $[\ell-2k, 2m-\ell-2k-3]$. Apply reducing formulae of (7) to these expressions, we have

$$\begin{aligned} x^{\ell-2k} &= x^{m+\ell-2k} + x^{\ell-k}, \\ &\vdots \\ x^{-k-1} &= x^{m-2k-1} + x^{-1}, \\ x^{m-k} &= x^0 + x^{-k}, \\ x^{m-k+1} &= x^1 + x^{-k+1}, \\ &\vdots \\ x^{2m-\ell-2k-3} &= x^{m-\ell-k-3} + x^{m-\ell-2k-3}. \end{aligned}$$

The exponents of x in the right side now are all in the range $[-k, m-k-1]$, no further reduction is needed. □

B.4 Proof of Proposition 3

Proof For simplicity, we combine the proof of case 3 and 6 together.

Case 3 and 6: In these cases, as $\ell \ge k$ and $\Delta_i = (n-2i+1)\ell - k$ (n even), $\Delta_i = (n-2i)\ell - k$ (n odd), we have all the Δ_is are greater than 0. Therefore, the modular reduction of $p_0^{(i)} x^{\Delta_i - k}$ is given by:

$$\begin{aligned} p_0^{(i)} x^{\Delta_i-k} \bmod f(x) = &\sum_{j=0}^{m-1} h_j^{(i)} x^{-k+(j+\Delta_i) \bmod m} \\ &+ \sum_{j=m-\Delta_i}^{m-1} h_j^{(i)} x^{j+\Delta_i-m}, \end{aligned} \tag{21}$$

for $i = 1, 2, \cdots, \lambda$, $\lambda = \frac{n}{2}$ if n is even, and $\lambda = \frac{n-1}{2}$ if n odd.

Meanwhile, it is easy to check that $p_1^{(i)} x^{\Delta_i}, p_1^{(i)} x^{\Delta_i - k}$ needs no reduction any more. We also note that $\Delta_i - \Delta_{i+1} = 2\ell$ for $i = 1, 2, \cdots, \lambda-1$ and $p_1^{(i)}$s consist of at most $\ell + r - 2$ terms. Thus, there are no overlapped terms among $p_1^{(i)} x^{\Delta_i}$ and $p_1^{(j)} x^{\Delta_j}$ if $i \ne j$. Two independent expressions $\sum_{i=1}^{\lambda} p_1^{(i)} x^{\Delta_i}$ and $\sum_{i=1}^{\lambda} p_1^{(i)} x^{\Delta_i - k}$ can be implemented in parallel. Plus n expressions in (21), we immediately conclude the proposition.

Furthermore, if $0 \le r \le 2$, n is even and $\ell \ge k$, we have $g_i = p_0^{(i)}, p_0^{(i)}$. In this case, the modular reduction of $g_i x^{\Delta_i - k}$ can be obtained using only (21). Obviously, this case also obeys this proposition.

Case 4: In this case, we note that $\ell < k, (n-2)\ell \le k$. In fact, one can check that

$$\begin{aligned} &(n+1)\ell > m = n\ell + r \ge 2k \\ \Rightarrow\quad &\tfrac{(n+1)\ell}{2} > k. \end{aligned}$$

But if $n \ge 5$, we have $(n-2)\ell \ge \frac{(n+1)\ell}{2} > k$. Therefore, Case 4 only happens if $n = 3$. In this case, all the $D_{s,t}$s constitute to an integral g_1. Now, we have

$$\begin{aligned} S_2 x^{-2k} \bmod f(x) &= g_1 x^{\ell-2k} \bmod f(x) \\ &= (p_1^{(1)} + p_1^{(1)} x^{-k} + p_0^{(1)} x^{-k}) x^{\ell-k}. \end{aligned}$$

Obviously, the modular reduction of above subexpressions are given by:

$$p_0^{(1)}x^{\ell-2k} \bmod f(x) = \sum_{j=0}^{m-1} h_j^{(1)} x^{-k+(j+\ell-k) \bmod m} + \sum_{j=0}^{k-\ell-1} h_j^{(1)} x^{j+\ell-k}, \tag{22}$$

and

$$\begin{aligned} & p_1^{(1)}x^{\ell-2k} \bmod f(x) \\ &= \left(p_{1,1}^{(1)}x^{k-\ell} + p_{1,2}^{(1)}\right) x^{\ell-2k} \bmod f(x) \\ &= p_{1,1}^{(1)}x^{-k} + p_{1,2}^{(1)}(x^{m+\ell-2k} + x^{\ell-k}). \end{aligned} \tag{23}$$

Specifically, no reduction is needed for $p_1^{(1)}x^{\ell-k}$, as all its term degrees are in the range $[-k, m-k-1]$. Combining it with (23), we have

$$\begin{aligned} & p_1^{(1)}x^{\ell-k} + p_{1,1}^{(1)}x^{-k} + p_{1,2}^{(1)}(x^{m+\ell-2k} + x^{\ell-k}) \\ &= p_{1,2}^{(1)}x^{m+\ell-2k} + p_{1,1}^{(1)}(x^{\ell-k} + x^{-k}). \end{aligned} \tag{24}$$

We directly know that (24) and (22) contains five subexpressions, which cost at most $\lceil \log_2(3+2) \rceil = \lceil \log_2 5 \rceil T_X$ in parallel.

Case 5: The proof of this case is analogous with that of Case 2. Recall that in this case $\Delta_i = (n-2i)\ell - k, i = 1, 2, \cdots, \frac{n-1}{2}$. Let an odd integer $t \geq 1$ satisfy that $t\ell \leq k, (t+2)\ell > k$. Then, we have $\Delta_i > 0$, for $i = 1, 2, \cdots, \frac{n-t}{2} - 1$ and $\Delta_i \leq 0$ for $i = \frac{n-t}{2}, \cdots, \frac{n-1}{2}$. Thus, if $i = 1, 2, \cdots, \frac{n-t}{2} - 1$, the modular reduction of $p_0^{(i)}x^{\Delta_i - k}$ is the same as (13), while if $i = \frac{n-t}{2}, \cdots, \frac{n-1}{2}$, its modular reduction is the same as (14).

Note that $p_1^{(i)} = \sum_{j=0}^{\ell+r-3} h_{m+j}^{(i)} x^j$. It is clear that the degrees of $p_1^{(i)}x^{\Delta_i}$ are all in the range $[-k, m-k-1]$, which need no reduction. On top of that, the explicit reduction of $p_1^{(i)}x^{\Delta_i - k}$ are given by

$$p_1^{(i)}x^{\Delta_i - k} \bmod f(x) = p_1^{(i)}x^{m+\Delta_i-k} + p_1^{(i)}x^{\Delta_i},$$

for $i = \frac{n-t}{2} + 1, \cdots, \frac{n-1}{2}$. Meanwhile,

$$\begin{aligned} & p_1^{(\frac{n-t}{2})}x^{t\ell-2k} \bmod f(x) \\ &= \left(p_{1,1}^{(\frac{n-t}{2})}x^{k-t\ell} + p_{1,2}^{(\frac{n-t}{2})}\right) x^{t\ell-2k} \bmod f(x) \\ &= p_{1,1}^{(\frac{n-t}{2})}x^{-k} + p_{1,2}^{(\frac{n-t}{2})}(x^{m+t\ell-2k} + x^{t\ell-k}). \end{aligned}$$

Here, $p_{1,1}^{(\frac{n-t}{2})}$ consists of at most $\ell + r - 2 - (k - t\ell)$ bits and $p_{1,2}^{(\frac{n-t+1}{2})}$ consists of at most $k - t\ell$ bits.

As a result, the modular reduction related to $\sum_{i=1}^{\frac{n-1}{2}}(p_1^{(i)} + p_1^{(i)}x^{-k})x^{\Delta_i}$ can be rewritten as two parts:

$$\begin{aligned} & \sum_{i=1}^{\frac{n-1}{2}} p_1^{(i)}x^{\Delta_i} + \sum_{i=\frac{n-t}{2}+1}^{\frac{n-1}{2}} p_1^{(i)}x^{\Delta_i} + p_{1,1}^{(\frac{n-t}{2})}x^{-k} + p_{1,2}^{(\frac{n-t}{2})}x^{t\ell-k} \\ &= \sum_{i=1}^{\frac{n-t}{2}-1} p_1^{(i)}x^{\Delta_i} + p_{1,1}^{(\frac{n-t}{2})}(x^{t\ell-k} + x^{-k}), \end{aligned} \tag{25}$$

$$\sum_{i=\frac{n-t}{2}+1}^{\frac{n-1}{2}} p_1^{(i)}x^{m+\Delta_i-k} + \sum_{i=1}^{\frac{n-t}{2}-1} p_1^{(i)}x^{\Delta_i-k} + p_{1,2}^{(\frac{n-t}{2})}x^{m+t\ell-2k}, \tag{26}$$

Similar with Case 2, one can easily check that the subexpressions in (25) and (26) have no overlapped terms with each other. So that they can be concatenated together without any logic gates. Meanwhile, $\sum_{i=1}^{\frac{n-1}{2}} p_0^{(i)}x^{\Delta_i - k} \bmod f(x)$ consists of at most $2 \cdot \frac{n-1}{2} = n-1$ subexpressions, and some of these subexpressions have no overlapped term with each other. It totally requires $\lceil \log_2(\frac{n-1}{2} + 2 + \max\{\frac{n-t}{2} - 1, \frac{t+1}{2}\}) \rceil \leq \lceil \log(n+2) \rceil T_X$ delay in parallel. □

Yin Li is an associated professor in Department of Computer Science and Technology, Xinyang Normal University, Henan, China. He received his BSc degree and MSc degrees from Information Engineering University, Zhenzhou, in 2004 and 2007, and the PhD in Computer Science from Shanghai Jiaotong University (SJTU), Shanghai (2011). He was a postdoc at Ben-Gurion University of the Negev, Israel, assisted by Prof. Shlomi Dolev. His current research interests include algorithm and architectures for computation in finite field, computer algebra, and secure cloud computing.

Shantanu Sharma received his PhD in Computer Science in 2016 from Ben-Gurion University, Israel, and Master of Technology (M.Tech.) degree in Computer Engineering from National Institute of Technology, Kurukshetra, India, in 2011. He was awarded a gold medal for the first position in his M.Tech. degree. During his PhD, he worked with Prof. Shlomi Dolev and Prof. Jeffrey Ullman. Currently, he is pursuing his Post Doc at the University of California, Irvine, USA, assisted by Prof. Sharad Mehrotra. His research interests include data security/privacy, designing models for MapReduce computations, distributed algorithms, mobile computing, and wireless communication.

Yu Zhang received the BSc degree in Henan university of economics and law in 2008, and the PhD degree in Huazhong university of science and technology in 2015. Since 2016, he has worked in the Department of computer science and information technology, Xinyang Normal University, where he is now a lecturer. His major interests include information security, cryptography, and information retrieval.

Xingpo Ma received his MSc degree from Central South University in China in 2005 with the first rank of electronic and information engineering, and received his PhD degree in computer application technology from the same university in 2013. Since 1st July 2014, he has been working in Xinyang Normal University in China. He was awarded the title of the urgently-needed talent in IOT by the Department of Industry and Information of China in 2014, and the youth backbone teacher by Xinyang Normal University in 2015. He is now the member of Chinese Association of Automation. His research interests include 5G networking and security of IOT.

Chuanda Qi received BSc degree in mathematics from Xinyang Normal University in 1985, and the PhD degree in cryptography from Information Engineering University in 2007. He is currently a professor in the Department of Computer Science and Technology at Xinyang Normal University, Xinyang. His research interests include cryptography, complexity theory, and mathematical logic.