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

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

I. Introduction

There is an increasing attention about the fast arithmetic operations in finite field $GF(2^m)$, which have many applications such as coding theory and cryptography [1], [3]. Specifically, efficient arithmetic algorithms and their related hardware architectures are crucial to high performance of these applications. Among the arithmetic operations defined in $GF(2^m)$, field multiplication is one of the most frequently desired operations, as other complex operations such as 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, the field multiplication consists of a polynomial multiplication followed by a modular reduction. Generally speaking, there are three types of bit-parallel $GF(2^m)$ multipliers of different architectures, i.e., quadratic [12], [13], [20], [21], subquadratic [6], [5] and hybrid bit-parallel multipliers [10], [8], [19], [9], [14]. Quadratic multipliers normally utilize schoolbook 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 bigger than quadratic or hybrid counterparts. Conversely, the hybrid multipliers can offer a trade-off between the time and space complexities [10]. 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 [17], [28], where T_X is the delay of one 2-input XOR gate. 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 Weimerskirch and Paar [6] and 4, 5 and 6-term of Karatsuba-like formulae introduced by Montgomery [4]. 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 [7]. Compared with 2-term KA, all these variations can obtain even fewer coefficients multiplications. However, these Karatsuba-like formulae usually contain complicated linear combinations of the split parts, which will yield extra gates delay for the bit-parallel multiplier. Conversely, Weimerskirch and Paar's approach is more fit for constructing hybrid $GF(2^m)$ multipliers, as all the intermediate inputs can be obtained in one T_X delay[29]. We call this algorithm as n-term KA and use this notion thereafter.

Recently, Li et al. investigated the application of n-term KA to a special class of trinomial $x^m + x^k + 1, m = nk$, and developed an efficient hybrid multiplier [28]. The lower bound of the 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 [17]. Nevertheless, such irreducible trinomials are not abundant, so that puts a confinement to the application of n-term KA. Park et al. [30] generalized above scheme and investigated the n-term Karatsuba hybrid multipliers for $x^m + x^k + 1$, where $m = n\ell$ or $m = n\ell + 1$. Inspired by this point, in this paper, we focus on an extension of the schemes in [28] and [30]. We investigate the application of n-term KA over general trinomials, i.e., $x^m + x^k + 1, m > 2k$. Note that m may not be divisible by n. The degree m is generally partitioned as $m = n\ell + r$ with $r < n, r < \ell$. We also use shifted polynomial basis (SPB) to optimize the modular reduction. Specifically, since the polynomial multiplication is partitioned into several independent parts and computed in parallel, constructing Mastrovito matrices for all these parts becomes more complicated. We utilize an alternative approach to perform modular reduction and exploit spatial correlation among different subexpressions. The main architecture is described in details. We explicitly study the space and time complexities under different parameters n, ℓ and r. Some upper and lower bounds with respect to these complexities are evaluated. As a result, we show that the optimal space complexity of our proposal is $m^2/2 + O(\sqrt{11}m^{3/2}/4)$, which roughly matches the best result of [28], [30]. 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.

Yin Li, Yu Zhang, Xingpo Ma and Chuanda Qi are with 1. Department of Computer Science and Technology, Xinyang Normal University, 2. Henan Key Lab of Analysis and Application of Educating Big Data, Henan, P.R.China, 464000. Shantanu Sharma is with Department of Computer Science at the University of California, Irvine, USA.

email: yunfeiyangli@gmail.com (Yin Li).

The rest of our paper is organized as follows: In Section 2, we briefly review a n-term KA formula and some relevant notions. Then, we investigate the application of n-term KA for polynomial multiplication of arbitrary degrees in Section 3. A new bit-parallel multiplier architecture is proposed as well. Section 4 presents an analysis of our proposal and study the optimal parameters of n-term KA and irreducible trinomials. Finally, some conclusions are drawn.

II. Preliminary

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

A. Shifted Polynomial Basis

The shifted polynomial basis (SPB) was originally proposed by Fan and Dai [11] 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 *[11] 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 $x^{-v}M := \{x^{i-v} | 0 \le i \le m-1\}$ 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 [12]. Specially, for trinomial $x^m + x^k + 1$, it has been proved that the optimal value of v is k or $k-1$ [11]. In this paper, we choose that $v = k$ and use this denotation thereafter. Additionally, we utilize both SPB and PB reduction in our scheme, and the default modular reduction refers the SPB one without specification.

B. n-term Karatsuba Algorithm

Besides 2-term Karatsuba algorithm, Weimerskirch and Paar [6] gave a generalized Karatsuba formulae, say n-term KA, that is applicable for the polynomial multiplication of arbitrary degree. Firstly, assume that there are two n-term polynomial with $n-1$ degree over $\mathbb{F}_2$:

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

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

$$D_i = a_i b_i.$$

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

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

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

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

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

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

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 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 [4] and Fan [7] 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.

III. Bit-parallel Multiplier using n-term Karatsuba Algorithm

In this section, we firstly investigate the multiplication of two m-term polynomials using n-term KA ($m \geq n$). Then, the modular reduction for related results are considered. Accordingly, we propose an efficient bit-parallel n-term Karatsuba multiplier architecture.

Provide that $f(x) = x^m + x^k + 1$ be an irreducible trinomial that defines the finite field $GF(2^m)$. Without loss of generality, we only consider the case of $m \geq 2k$, as the reciprocal polynomial $x^m + x^{m-k} + 1$ is also irreducible whenever $x^m + x^k + 1$ is irreducible. 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.,

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

A. Polynomial multiplication using n-term Karatsuba algorithm

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:*

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

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

$$\delta_{s,t} = \begin{cases} s + t - 2(n-r), & \text{if } s > t > n - r, \\ s - (n-r), & \text{if } s > n - r, t \leq n - r, \\ 0, & \text{if } 0 < t < s \leq n - r. \end{cases} \tag{3}$$

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

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

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

$$\begin{aligned} AB &= \sum_{i=0}^{n-1} A_iB_{n-1}x^{\overline{i}+\overline{n-1}} + \cdots + \sum_{i=0}^{n-1} A_iB_0x^{\overline{i}+\overline{0}} \\ &= \sum_{i=0}^{n-1} A_iB_ix^{2\overline{i}} + \sum_{0\le i<j<n} (A_iB_j + A_jB_i)x^{\overline{i}+\overline{j}} \end{aligned} \tag{4}$$

Applying Eq. (1), we know that $(A_iB_j + A_jB_i)x^{\overline{i}+\overline{j}} = ((A_i + A_j)(B_i + B_j) + A_iB_i + A_jB_j)x^{\overline{i}+\overline{j}}$. Plug these formulae into above expression, Eq. (4) can be rewritten as:

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

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

Analogous to the approach present in [17], we can divide (2) into two parts and compute them independently, i.e.,

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

Therefore, the SPB field multiplication is given by

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

In the following subsections, we discuss the computation of S_1x^{-2k}, S_2x^{-2k} and analyze their complexities, respectively.

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

Since

$$S_1 = \left(\sum_{i=n-r+1}^{n-1} A_iB_ix^{(\ell+1)i-n+r} + \sum_{i=0}^{n-r} A_iB_ix^{\ell i} \right) h(x),$$

we firstly consider the calculation of the subexpression in the parenthesis, denoted by $E(x)$, and then S_1x^{-2k} modulo $f(x)$.

In fact, one can compute S_1x^{-2k} modulo $f(x)$ by constructing a Mastrovito matrix with respect to S_1x^{-2k}, analogous to the authors did in [28], [30]. Nevertheless, we found that if the irreducible trinomial $x^m + x^k + 1$ is not special, e.g., $m = nk$, related Mastrovito matrix for 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 computes $E(x)$ first and then S_1x^{-2k}. This approach increases the time delay a little but can save more logic gates.

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

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

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

$$e_i = \begin{cases} c_i^{(0)} & 0 \le i \le \ell-1, \\ c_i^{(0)} + c_{i-\ell}^{(1)} & \ell \le i \le 2\ell-2, \\ c_{i-\ell}^{(1)} & i = 2\ell-1, \\ c_{i-\ell}^{(1)} + c_{i-2\ell}^{(2)} & 2\ell \le i \le 3\ell-2, \\ \quad\vdots & \\ c_{i-(n-r-1)\ell}^{(n-r-1)} + c_{i-(n-r)\ell}^{(n-r)} & (n-r)\ell \le i \le (n-r+1)\ell-2, \\ c_{i-(n-r)\ell}^{(n-r)} & i = (n-r+1)\ell-1, (n-r+1)\ell \\ \quad\vdots & \\ c_{i-(n-2)\ell-r+2}^{(n-2)} + c_{i-(n-1)\ell-r+1}^{(n-1)} & (n-1)\ell+r-1 \le i \le m-2, \\ c_{i-(n-1)\ell-r+1}^{(n-1)} & m-1 \le i \le m+\ell-1. \end{cases} \tag{5}$$

Recall that $\deg(E) = m+\ell-1$. If we perform the PB reduction $E(x) \bmod f(x)$, there are only ℓ terms of $E(x)$ that needs to be reduced further. To save the circuit delay of implementation, we can compute $E(x) \bmod f(x)$ directly, which can be obtained by using the formula $x^i = x^{i-m} + x^{i-m+k}$. Provide that $E(x) = p_1 x^m + p_0$, where $p_1(x) = \sum_{i=0}^{\ell-1} e_{i+m} x^i$ and $p_0(x) = \sum_{i=0}^{m-1} e_i x^i$. Then, we have

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

Let $E'(x) = \sum_{i=0}^{m-1} e'_i x^i$ 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.,

$$e'_i = \begin{cases} c_i^{(0)} + c_{i+\ell+1}^{(n-1)} & 0 \le i \le \ell-1, \\ c_i^{(0)} + c_{i-\ell}^{(1)} & \ell \le i \le 2\ell-2, \\ c_{i-\ell}^{(1)} & i = 2\ell-1, \\ c_{i-\ell}^{(1)} + c_{i-2\ell}^{(2)} & 2\ell \le i \le 3\ell-2, \\ \quad\vdots & \\ c_{i-(n-r-1)\ell}^{(n-r-1)} + c_{i-(n-r)\ell}^{(n-r)} & (n-r)\ell \le i \le (n-r+1)\ell-2, \\ c_{i-(n-r)\ell}^{(n-r)} & i = (n-r+1)\ell-1, (n-r+1)\ell, \\ \quad\vdots & \\ c_{i-(n-2)\ell-r+2}^{(n-2)} + c_{i-(n-1)\ell-r+1}^{(n-1)} & (n-1)\ell+r-1 \le i \le m-2, \\ c_{i-(n-1)\ell-r+1}^{(n-1)} & i = m-1. \end{cases} \tag{6}$$

Thus, one can calculate (6) instead of (5), and the computation of p_0 can be combined with that of (6). We give the details in Section 4.1.

After that, we then consider the SPB modular reduction of $S_1 x^{-2k}$. Note that

$$\begin{aligned} S_1 x^{-2k} \bmod f(x) &= E(x)h(x)x^{-2k} \bmod f(x) \\ &= [p_1 x^k + (p_1+p_0)]h(x)x^{-2k} \bmod f(x) \\ &= E'(x)h(x)x^{-2k} + p_1(x)h(x)x^{-k} \bmod f(x). \end{aligned}$$

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 $\epsilon_i > 0$ 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 $p_1 x^{i\ell+\epsilon_i}$ and $p_1 x^{(i-1)\ell+\epsilon_{i-1}}$, for $i = 1, 2, \cdots, n-1$. Also, one can check that $\deg(p_1 h x^{-k}) = (n-1)\ell + r - 1 + \ell - 1 - k = m-k-2$, 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. Figure 1 depicts bit positions for these subexpressions.

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

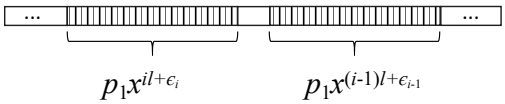

Fig. 1. Bit positions for $p_1 x^{i\ell+\epsilon_i}, i = 1, 2, \cdots, n-1$.

operations depend on the magnitude relations between k and ℓ. Recall that $k \leq m/2$, and $m = n\ell + r, n > r, \ell > r$. Two cases are considered:

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

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

$$\begin{aligned} &n\ell \geq 3\ell > 2\ell + r \Rightarrow (n-1)\ell > \ell + r \\ &\Rightarrow 2(n-1)\ell > n\ell + r = m \Rightarrow (n-1)\ell > m/2 \geq k. \end{aligned}$$

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 [17]. Thus, we only analyze the case of $k < (n-1)\ell$ in this study. The SPB reduction relies on the following formula:

$$\begin{cases} x^i = x^{m+i} + x^{i+k}, \text{ for } i = -2k, \cdots, -k-1; \\ x^i = x^{i-m} + x^{i-m+k}, \text{ for } i = m-k, m-k+1, \\ \qquad\qquad\qquad\qquad\qquad \cdots, 2m-2k-2. \end{cases} \tag{7}$$

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*

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

The proof of above lemma can be found in the appendix. 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:

$$\begin{aligned} E'(x)x^{i\ell+\epsilon_i-2k} \bmod f(x) = &\sum_{j=0}^{m-1} e'_j x^{-k+(j+\theta_i) \bmod m} \\ &+ \sum_{j=0}^{-\theta_i-1} e'_j x^{j+\theta_i}, \end{aligned} \tag{8}$$

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

$$\begin{aligned} E'(x)x^{i\ell+\epsilon_i-2k} \bmod f(x) = &\sum_{j=0}^{m-1} e'_j x^{-k+(j+\theta_i) \bmod m} \\ &+ \sum_{j=m-\theta_i}^{m-1} e'_j x^{j+\theta_i-m}, \end{aligned} \tag{9}$$

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 (8) with $\theta_i = 0$ and $\sum_{j=0}^{-\theta_i-1} d_j x^{j+\theta_i} = 0$. For simplicity, we do not discuss this case independently.

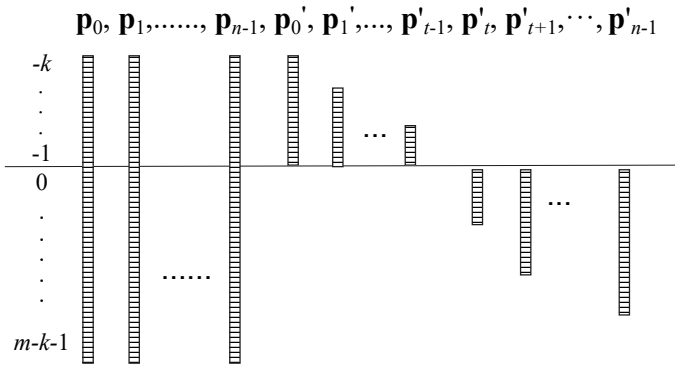

Fig. 2. Bit positions for all the subexpressions.

One can easily check that (8) and (9) consist of two subexpressions, in which the former one contains m terms and the latter one contains θ_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. Figure 2 demonstrates the bit positions for these subexpressions.

The vectors $\mathbf{P}_i, \mathbf{P}'_i$ in Figure 2 represent the coefficients vectors with respect to all the subexpressions presented in (8) and (9). 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.

C. 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 a 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

$$D_{s,t} = (\sum_{i=0}^{\ell-1} u_i^{(s,t)}) \cdot (\sum_{i=0}^{\ell-1} v_i^{(s,t)}) = \sum_{i=0}^{2\ell-2} d_i^{(s,t)} x^i, \tag{10}$$

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

$$D_{s,t} = (\sum_{i=0}^{\ell} u_i^{(s,t)}) \cdot (\sum_{i=0}^{\ell} v_i^{(s,t)}) = \sum_{i=0}^{2\ell} d_i^{(s,t)} x^i, \tag{11}$$

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 [27] 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*

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

or

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

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 [27]. One just replaces $D_{s,t}$ by $D_{s,t} \cdot x^{\delta_{s,t}}$ in that proof and obtains the conclusion directly.

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

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 (3), 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, $h_{(s-1)\ell+\delta}^{(i)}$ is a plus of three terms. Except this case, there is no coefficient of g_i obtained by a plus of three terms. Plug (10) and (11) 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.

2) Step (iii) and (iv): Then we consider the computations of Step (iii) and (iv). Firstly we have a following observation.

Observation 1 The modular reduction of $g_1x^{(2\lambda-1)\ell-2k}$, $g_2x^{(2\lambda-3)\ell-2k}$, $\cdots, g_\lambda x^{\ell-2k}$ by $f(x)$ only require one reduction step.

The proof of this observation is given in the appendix. 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 $g_1x^{(2\lambda-1)\ell-2k}, g_2x^{(2\lambda-3)\ell-2k}, \cdots, g_\lambda x^{\ell-2k}$ 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 $g_ix^{\Delta_i-k}$ 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 $g_ix^{\Delta_i-k}$ 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 [17]; 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.,

$$g_i = p_1^{(i)}x^m + p_0^{(i)}, \tag{12}$$

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 $g_ix^{\Delta_i-k}$ can be expressed as the reduction with respect to $p_1^{(i)}, p_2^{(i)}$ multiplying certain exponent of x. More explicitly,

$$\begin{aligned} g_ix^{\Delta_i-k} \bmod f(x) = \Big(&p_1^{(i)} + p_1^{(i)}x^{-k} + \\ &p_0^{(i)}x^{-k}\Big)\, x^{\Delta_i} \bmod f(x), \end{aligned} \tag{13}$$

$i=1,2,\cdots,\lambda$. Consider the term degree range of SPB representation, 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 $g_ix^{\Delta_i-k}$ 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 $\sum_{i=1}^{\lambda} g_ix^{(2\lambda-2i+1)\ell-2k} \bmod f(x)$, 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 the appendix.

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 (13). Note that $p_1^{(i)}=\sum_{j=0}^{r-3}h_{m+j}^{(i)}x^j$ and $p_2^{(i)}=\sum_{j=0}^{m-1}h_j^{(i)}x^j$ here. Firstly, the modular reduction of $p_2^{(i)}x^{\Delta_i-k}$ can be obtained as follows:

$$\begin{aligned} p_2^{(i)}x^{\Delta_i-k} \bmod f(x) = &\sum_{j=0}^{m-1} h_j^{(i)}x^{-k+(j+\Delta_i) \bmod m} \\ &+ \sum_{j=m-\Delta_i}^{m-1} h_j^{(i)}x^{j+\Delta_i-m}, \end{aligned} \tag{14}$$

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

$$p_2^{(i)} x^{\Delta_i - k} \bmod f(x) = \sum_{j=0}^{m-1} h_j^{(i)} x^{-k+(j+\Delta_i) \bmod m} + \sum_{j=0}^{-\Delta_i - 1} h_j^{(i)} x^{j+\Delta_i}, \tag{15}$$

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,

$$\sum_{i=1}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i} \bmod f(x) = \sum_{i=1}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i}. \tag{16}$$

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 $t\ell < k$ and $t\ell + r - 3 \geq k$, 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

$$p_1^{(i)} x^{\Delta_i - k} \bmod f(x) = p_1^{(i)} x^{m + \Delta_i - k} + p_1^{(i)} x^{\Delta_i}, \tag{17}$$

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

$$\begin{aligned} & p_1^{(\frac{n-t+1}{2})} x^{t\ell - 2k} \bmod f(x) \\ &= \left(p_{1,1}^{(\frac{n-t+1}{2})} x^{k - t\ell} + p_{1,2}^{(\frac{n-t+1}{2})} \right) x^{t\ell - 2k} \bmod f(x) \\ &= p_{1,1}^{(\frac{n-t+1}{2})} x^{-k} + p_{1,2}^{(\frac{n-t+1}{2})} (x^{m + t\ell - 2k} + x^{t\ell - k}). \end{aligned} \tag{18}$$

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 (16), (17) and (18), 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:

$$\begin{aligned} & \sum_{i=1}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i} + \sum_{i=\frac{n-t+1}{2}}^{\frac{n}{2}} p_1^{(i)} x^{\Delta_i} + p_{1,1}^{(\frac{n-t+1}{2})} x^{-k} + p_{1,2}^{(\frac{n-t+1}{2})} x^{t\ell - k} \\ &= \sum_{i=1}^{\frac{n-t-1}{2}} p_1^{(i)} x^{\Delta_i} + p_{1,1}^{(\frac{n-t+1}{2})} (1 + x^{-k}), \end{aligned} \tag{19}$$

$$\sum_{i=\frac{n-t+3}{2}}^{\frac{n}{2}} p_1^{(i)} x^{m + \Delta_i - k} + \sum_{i=1}^{\frac{n-t-1}{2}} p_1^{(i)} x^{\Delta_i - k} + p_{1,2}^{(\frac{n-t+1}{2})} x^{m + t\ell - 2k}, \tag{20}$$

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

Finally, we add all the modular reduction results included in (14), (15), (19) and (20) 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 (14) does not overlap with $\sum_{j=0}^{-\Delta_i - 1} h_j^{(i)} x^{j + \Delta_i}$ in (15), 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.

D. A small example of n-term Karatsuba multiplier

To illustrate the n-term Karatsuba algorithm and the modular reduction strategy related to S_1x^{-2k} and S_2x^{-2k}, we give a small example. Consider the field multiplication using SPB representation over $GF(2^{14})$ with the underlying irreducible trinomial $x^{14}+x^5+1$. Obviously, we have the optimal SPB parameter $k=5$ and SPB is defined as $\{x^{-5},x^{-4},\cdots,x^7,x^8\}$. Provide that $A\cdot x^{-5}=\sum_{i=0}^{13}a_ix^{i-5}$ and $B\cdot x^{-5}=\sum_{i=0}^{13}b_ix^{i-5}$ are two elements in $GF(2^{14})$ in SPB representation.

Without loss of generality, we apply 4-term Karatsuba algorithm to the polynomial multiplication. It is clear that $14=4\times3+2$. We have $n=4,\ell=3,r=2$ and r satisfies $r<n,r<\ell$. Accordingly, partition A,B as $A=A_3x^{10}+A_2x^6+A_1x^3+A_0, B=B_3x^{10}+B_2x^6+B_1x^3+B_0$, where

$$A_i=\sum_{j=0}^{2}a_{j+3i}x^j, B_i=\sum_{j=0}^{2}a_{j+3i}x^j,\ \text{for } i=0,1,$$
$$A_i=\sum_{j=0}^{3}a_{j+4i-2}x^j, B_i=\sum_{j=0}^{3}a_{j+4i-2}x^j,\ \text{for } i=2,3.$$

From equation (2),

$$\begin{aligned}AB=&(A_3B_3x^{10}+A_2B_2x^6+A_1B_1x^3+A_0B_0)h(x)+D_{3,2}x^{15+1}\\&+D_{3,1}x^{12+1}+D_{3,0}x^{9+1}+D_{2,1}x^9+D_{2,0}x^6+D_{1,0}x^3\\=&S_1+S_2,\end{aligned}$$

where $h(x)=x^{10}+x^6+x^3+1$. Recall that $D_{s,t}=(A_s+A_t)(B_s+B_t)$. Apparently, there are $\binom{5}{2}=10$ such $D_{s,t}$s. We also write the term degrees of $D_{s,t}$ in the form of $i\ell+\delta_{s,t}$, in which $\delta_{3,2}=\delta_{3,1}=\delta_{3,0}=1$ and the rest of $\delta_{s,t}$ are all zero.

Based on the formulations of A_i,B_i, we assume that $D_{s,t}=\sum_{i=0}^{6}d_i^{(s,t)}x^i$ for $3\geq s>1,2\geq t>0,s\neq t$ and $D_{1,0}=\sum_{i=0}^{4}d_i^{(1,0)}x^i$. According to the description in Section 3.1, we have

$$S_1=(A_3B_3x^{10}+A_2B_2x^6+A_1B_1x^3+A_0B_0)h(x),$$

and categorize S_2 into to two parts, i.e., $S_2=g_1x^9+g_2x^3$, where

$$\begin{aligned}g_1&=D_{3,2}x^{6+1}+D_{3,1}x^{3+1}+D_{3,0}x,\\g_2&=D_{2,1}x^6+D_{2,0}x^3+D_{1,0}.\end{aligned}$$

Now, consider the modular reduction of S_1x^{-10} and S_2x^{-10}. We first compute $E(x)=A_3B_3x^{10}+A_2B_2x^6+A_1B_1x^3+A_0B_0=p_1x^m+p_0$. Obviously,

$$\begin{aligned}p_0=&(a_{13}b_{10}+a_{12}b_{11}+a_{11}b_{12}+a_{10}b_{13})x^{13}+(a_{12}b_{10}+a_{11}b_{11}\\&+a_{10}b_{12}+a_9b_9)x^{12}+(a_{11}b_{10}+a_{10}b_{11}+a_9b_8+a_8b_9)x^{11}+\\&(a_{10}b_{10}+a_9b_7+a_8b_8+a_7b_9)x^{10}+(a_9b_6+a_8b_7+a_7b_8+a_6b_9)x^9\\&+(a_8b_6+a_7b_7+a_6b_8)x^8+(a_7b_6+a_6b_7+a_5b_5)x^7\\&+(a_6b_6+a_5b_4+a_4b_5)x^6+(a_5b_3+a_4b_4+a_3b_5)x^5\\&+(a_4b_3+a_3b_4+a_2b_2)x^4+(a_3b_3+a_2b_1+a_1b_2)x^3\\&+(a_2b_0+a_1b_1+a_0b_2)x^2+(a_1b_0+a_0b_1)x+a_0b_0,\end{aligned}$$

and $p_1=a_{13}b_{13}x^2+(a_{13}b_{12}+a_{12}b_{13})x+(a_{13}b_{11}+a_{12}b_{12}+a_{12}b_{13})$. Meanwhile,

$$\begin{aligned}g_1=&d_6^{(3,2)}x^{13}+d_5^{(3,2)}x^{12}+d_4^{(3,2)}x^{11}+\big(d_3^{(3,2)}+d_6^{(3,1)}\big)x^{10}\\&+\big(d_2^{(3,2)}+d_5^{(3,1)}\big)x^9+\big(d_1^{(3,2)}+d_4^{(3,1)}\big)x^8+\big(d_0^{(3,2)}+d_3^{(3,1)}\\&+d_6^{(3,0)}\big)x^7+\big(d_2^{(3,1)}+d_5^{(3,0)}\big)x^6+\big(d_1^{(3,1)}+d_4^{(3,0)}\big)x^5\\&+\big(d_0^{(3,1)}+d_3^{(3,0)}\big)x^4+d_2^{(3,0)}x^3+d_1^{(3,0)}x^2+d_0^{(3,0)}x,\\g_2=&d_6^{(2,1)}x^{12}+d_5^{(2,1)}x^{11}+d_4^{(2,1)}x^{10}+\big(d_3^{(2,1)}+d_6^{(2,0)}\big)x^9\\&+\big(d_2^{(2,1)}+d_5^{(2,0)}\big)x^8+\big(d_1^{(2,1)}+d_4^{(2,0)}\big)x^7+\big(d_0^{(2,1)}+d_3^{(2,0)}\big)x^6\\&+d_2^{(2,0)}x^5+\big(d_1^{(2,0)}+d_4^{(1,0)}\big)x^4+\big(d_0^{(2,0)}+d_3^{(1,0)}\big)x^3\\&+d_2^{(1,0)}x^2+d_1^{(1,0)}x^1+d_0^{(1,0)}.\end{aligned}$$

As presented in section 3.2, in order to speed up the multiplier, we can compute $E'(x)=p_1+p_0$ instead of p_0, while p_1 can be obtained by reusing some XOR gates of E'. On top of that, one can check that $p_1(x^{10}+x^6+x^3+1)x^{-5}$ has all its terms in the range $[-5,8]$, which does not need any further reduction. Then, we can easily obtain the reduction of $E'(x)h(x)x^{-10}$ and g_1x^{9-10},g_2x^{3-10} modulo $x^{14}+x^5+1$, which is available in the appendix.

TABLE I
THE COMPUTATION COMPLEXITY OF e'_i

e'_i	#AND	#XOR	Delay
$e'_0 = c_0^{(0)} + c_{\ell+1}^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$
$e'_1 = c_1^{(0)} + c_{\ell+2}^{(n-1)}$	$\ell+1$	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	$(n-r)\ell^2 + r(\ell+1)^2$	$(n-r)\ell(\ell-1) + r\ell(\ell+1)$	$T_A + (\lceil \log_2(\ell+1) \rceil) T_X$

IV. COMPLEXITY ANALYSIS

Based on previous description, in this section, we analyze the space and time complexities pertaining to $S_1 x^{-2k}$ and $S_2 x^{-2k}$ modulo $f(x)$.

A. Space and time complexity of $S_1 x^{-2k} \bmod f(x)$

As presented in section 3.2, the computation of $S_1 x^{-2k}$ 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 (5) and (6), 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 (5) and (6), it is easy to check that each coefficient e_i and e'_i are composed of at most $\ell+1$ coefficient products of $A_i B_i, i = 0, 1, \cdots, n-1$. 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 I 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 $p_1 + p_0$ 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* [16], [17]. 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 $a_i b_j$ 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.

TABLE II
SPACE AND TIME COMPLEXITIES OF $S_1x^{-2k} \bmod f(x)$.

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 III
NUMBER OF BITS IN (14), (15), (19) AND (20).

Formulae	number of bits
(14)	$m+\Delta_i, i=1,2,\cdots,\frac{n-t-1}{2}$
(15)	$m+\lvert\Delta_i\rvert, i=\frac{n-t+3}{2},\cdots,\frac{n}{2}$
(19)	$\frac{n-t-1}{2}\cdot(r-2)+2(r-2-(k-t\ell))$
(20)	$(\frac{n}{2}-1)\cdot(r-2)+(k-t\ell)$

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

B. Space and time complexity of $S_2x^{-2k} \bmod f(x)$

Now we discuss the complexity of $S_2x^{-2k} \bmod f(x)$ 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 $0<t<n-r$ 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 cost $(\ell+1)^2$ AND and ℓ^2 XOR gates. One also 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. So the additions among $D_{s,t}$s in the same category require $m+r-2-2\ell$ XOR gates (even n) or $m+r-2-\ell$ XOR gates (odd n).

In the end, as mentioned in Section 3.3, we need to add the modular results presented in (14), (15), (19) and (20) to obtain the final result. The number of required XOR gates depends on these formulations. For example, Table III presents the number of bits included in (14), (15), (19) and (20) for Case 2. It totally requires at most $\frac{m(n-1)}{2}+(n-\frac{t-1}{2})(r-2)+\sum_{i=1}^{n/2}|\Delta_i|$ XOR gates to add these expressions. Explicit space and time complexities for each steps are summarized in Table IV.

C. Total complexity and more discussion

As mentioned in previous sections, $S_1x^{-2k} \bmod f(x)$ and $S_2x^{-2k} \bmod f(x)$ are computed in parallel and the overall delay is equal to the longer circuit delay of either $S_1x^{-2k} \bmod f(x)$ or $S_2x^{-2k} \bmod f(x)$. From Table 2 and 3, it is clear that the

TABLE IV
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	$(\binom{n}{2} - \binom{n-r}{2})(\ell+1)^2$	$(\binom{n}{2} - \binom{n-r}{2})\ell^2$	
	Additions of $D_{s,t}$	-	$\frac{n}{2} \cdot (m + r - 2 - 2\ell)$ (even n) $\frac{n-1}{2} \cdot (m + r - 2 - \ell)$ (odd n)	
(iii), (iv)	Case 2	-	$\frac{m(n-2)}{2} + (n - \frac{t-1}{2})(r-2) + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert$	$\lceil \log_2(\frac{n+4}{2} + \max\{\frac{n-t-1}{2}, \frac{t+1}{2}\}) \rceil T_X$
	Case 3	-	$\frac{m(n-2)}{2} + n(r-2) + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert$	$\lceil \log_2(n+2) \rceil T_X$
	Case 4 ($n = 3$)	-	$m + 2\ell + 2r - 4$	$\lceil \log_2 5 \rceil T_X$
	Case 5	-	$\frac{(n-3)m}{2} + (n - \frac{t+1}{2})(r + \ell - 2) + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert$	$\lceil \log_2(\frac{n+5}{2} + \max\{\frac{n-t-2}{2}, \frac{t+1}{2}\}) \rceil T_X$
	Case 6	-	$\frac{(n-3)m}{2} + (n-1)(r + \ell - 2) + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert$	$\lceil \log_2(n+1) \rceil T_X$
where $\Delta_i = (n - 2i + 1)\ell - k$, if n is even or $(n - 2i)\ell - k$ if n is odd, $t \geq 1$ is an odd integer that satisfy $t\ell \leq k, (t+2)\ell > k$				

delay of $S_2 x^{-2k} \bmod f(x)$ is slightly higher. Thus the overall circuit 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 leads to one more T_X delay. To sum up, the total delay of our proposed architecture is

$$\text{Time Delay: } \leq T_A + (2 + \lceil \log_2(\ell+3) \rceil + \lceil \log_2(n+2) \rceil) T_X.$$

The space complexity is

$$\begin{aligned}
&\#\text{ AND: } \tfrac{m^2}{2} + \tfrac{m\ell}{2} + (m + n + \tfrac{\ell+1}{2})r - (\ell+2)r^2, \\
&\#\text{ XOR: } \tfrac{m^2}{2} + (2n + \tfrac{\ell}{2} + r - 2)m + \tfrac{n^2 + rn + r + \ell r}{2} + \textstyle\sum_{i=1}^{\ell} W(i) \\
&+ \textstyle\sum_{i=0}^{n-1} \lvert\theta_i\rvert + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert - \ell r^2 - \ell - \tfrac{7n}{2}, (n \text{ even}), \\
&\text{or} \\
&\tfrac{m^2}{2} + (2n + r + \tfrac{\ell-3}{2})m + \tfrac{n^2 + rn + \ell r + 6}{2} + \textstyle\sum_{i=1}^{\ell} W(i) \\
&+ \textstyle\sum_{i=0}^{n-1} \lvert\theta_i\rvert + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert - \ell r^2 - \tfrac{5r + 3\ell + 7n}{2}, (n \text{ odd}),
\end{aligned}$$

where the explicit values of Δ_i and θ_i are presented in Table 2 and 3. It is noteworthy that in Table 3, there are several cases for the number of required XOR gates. For simplicity, we only present the upper bound 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 $\frac{5m^2}{2}$.

Specifically, as r is a small integer, the functions related to mr can roughly be recognized as a linear function of m. 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 $\frac{\ell}{2} \log_2 \ell$ [16]. If we ignore these linear or small parts of above complexities formulae, the space complexity of our proposal is determined by some quadratic subexpressions.

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 IV, 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 $\sum_{i=0}^{n-1}|\theta_i|$ and $\sum_{i=1}^{\lambda}|\Delta_i|$ ($\lambda = n/2$ for even n and $\lambda = (n-1)/2$ of odd n) are determined by k. In fact,

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

and

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

where t satisfies $(t-1)\ell+\epsilon_{t-1} \le k < t\ell+\epsilon_t$ and t' is an odd integer satisfying $t'\ell \le k < (t'+2)\ell$. 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 [19] 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.

2) Optimal selection of n, ℓ: From previous description, we know that k highly influences the values of $\sum_{i=0}^{n-1}|\theta_i|$ and $\sum_{i=1}^{\lambda}|\Delta_i|$. 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 smaller than n, ℓ and 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. At this time, 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 Table V, we give a comparison of several different bit-parallel multipliers for irreducible trinomials. All these multipliers are using PB representations except particular description. It is clear that our scheme costs fewer logic gates than previous

architectures (quadratic or hybrid). The best of our result only costs about $m^2/2 + O(\sqrt{11}m^{3/2}/2)$ circuit gates. On the other hand, the time complexity of the proposed multiplier is slightly higher than the fastest result utilizing classic Karatsuba algorithm. In the following section, we investigate possible speedup strategy for our scheme under special type of trinomials.

V. 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

$$\begin{aligned} \lceil \log_2(\ell+3) \rceil + \lceil \log_2(n+2) \rceil &\leq 1 + \lceil \log_2(\ell+3)(n+2) \rceil \\ &= 1 + \lceil \log_2(m + 2\ell + 3n + 6) \rceil. \end{aligned}$$

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 [10]. In order to obtain a better space and time complexity trade-off, we want to apply a speedup strategy to our architecture, which was proposed in [28]. 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 [28], the authors utilized a special type of trinomial, where $f(x) = x^m + x^k + 1$ satisfies $m = n \cdot k$. 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 (5). Nevertheless, the subexpressions in (8) and (9) now have some common terms, which can save certain logic gates. More explicitly, since $k = t\ell$, we have

$$\begin{aligned} E'(x) x^{i\ell - 2t\ell} \bmod f(x) = & \underline{\sum_{j=(t-i)\ell}^{n\ell-1} e'_j x^{j+(i-2t)\ell}} \\ & + \sum_{j=0}^{(t-i)\ell-1} e'_j x^{j+(i+n-2t)\ell} + \sum_{j=0}^{(t-i)\ell-1} e'_j x^{j+(i-t)\ell}, \\ & \cdots\cdots\cdots\cdots\cdots\cdots \end{aligned} \tag{21}$$

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

$$\begin{aligned} E'(x) x^{i\ell - 2t\ell} \bmod f(x) = & \sum_{j=0}^{(n-i+t)\ell-1} e'_j x^{j+(i-2t)\ell} \\ & \cdots\cdots\cdots\cdots\cdots\cdots \\ & + \sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-2t-n)\ell} + \underline{\sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-n-t)\ell}}, \end{aligned} \tag{22}$$

for $i = t, t+1, \cdots, n-1$. Notice that if $i = t$, the subexpression $\sum_{j=(n-i+t)\ell}^{n\ell-1} e'_j x^{j+(i-n-t)\ell} = \sum_{j=n\ell}^{n\ell-1} e'_j x^{j-n\ell}$ 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 (21) and (22), 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,

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

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:

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

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 (21) can be eliminated by offsetting related expressions in (22), but only t solid underlined subexpressions in (22) can be eliminated. After combining the overlapped parts between (21) and (22), 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

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

without any logic gates. As a result, the addition between (21) and (22) 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 II, 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.

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 $n \geq 20$ and t approaches to $n/2$, the above inequation holds. 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 [28]. 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 (14), (15), (19) and (20). 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 reorganized the intermediate values of former steps. After applying speedup strategy, the circuit for the whole multiplier is

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

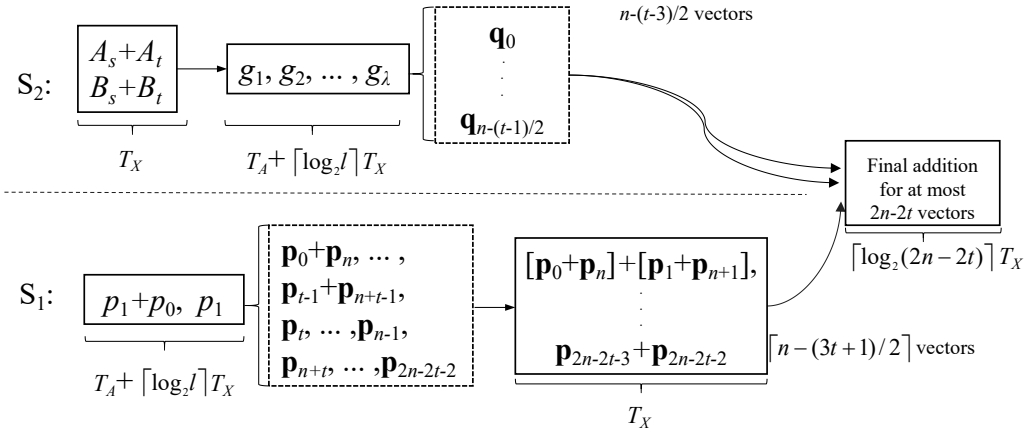

Fig. 3. Speedup Strategy for $x^{n\ell}+x^{t\ell}+1$.

which matches the results of [30]. Furthermore, since some subexpresssions in (21) and (22) can offset, certain number of XOR gates are saved. But this number is small, which has little effect on the overall space complexity. The study of optimal n, ℓ can follow the same line as we did in previous section.

TABLE V
COMPARISON OF SOME BIT-PARALLEL MULTIPLIERS FOR IRREDUCIBLE TRINOMIALS $x^m+x^k+1, m \geq 2k$

Multiplier	# AND	# XOR	Time delay
Montgomery[25], school-book[24] Mastrovito [20][21][22]	m^2	m^2-1	$T_A+(2+\lceil \log_2 m \rceil)T_X$
Mastrovito [23]	m^2	m^2-1	$T_A+(\lceil \log_2(2m+2k-3) \rceil)T_X$
SPB Mastrovito [12] Montgomery [13]	m^2	m^2-1	$T_A+\lceil \log_2(2m-k-1) \rceil T_X$
KA [10]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4}+4m+k-\frac{23}{4}$ (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[14]	$\frac{m^2}{2}+(m-k)^2$	$\frac{m^2}{2}+(m-k)^2+2k$	$T_A+(2+\lceil \log_2(m-1) \rceil)T_X$
Modified KA[9]	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[16]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4}+O(m\log_2 m)$ (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[26]	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 [17]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4}+\frac{m}{2}+O(m\log_2 m)$ (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 [28] 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 [30] $m=n\ell, n\ell+1$	$\frac{m^2}{2}+O(\frac{m\ell}{2})$	$\frac{m^2}{2}+\frac{m\ell}{2}+\frac{5mn}{4}+O(m\log_2 \ell)$	$\leq T_A+(2+\lceil \log_2 \ell \rceil+\lceil \log_2 n \rceil)T_X$
This paper (optimal) $m=n\ell+r$	$\frac{m^2}{2}+O\left(\frac{\sqrt{11}m^{3/2}}{4}\right)$	$\frac{m^2}{2}+O\left(\frac{\sqrt{11}m^{3/2}}{2}\right)$	$\leq T_A+(2+\lceil \log_2(\ell+3) \rceil$ $+\lceil \log_2(n+2) \rceil)T_X$
where $\Delta=m^2+\frac{(m-k)(m-1-3k)}{2}$ ($\frac{m-1}{3} \leq k < \frac{m}{2}$, $2^{v-1}<k\leq 2^v$), $\Theta=\max(3m-3k-1, 2m-2k+2^v)$			

VI. CONCLUSION

In this paper, we extend the application of a n-term Karatsuba algorithm for general trinomials x^m+x^k+1, by decomposing m into $m=n\ell+r$. Under such a decomposition, 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 $\frac{m^2}{2} + O(\frac{\sqrt{11}m^{3/2}}{4})$, which matches the best result of current hybrid multipliers. Meanwhile, its time complexity is slightly higher than the counterparts.

In addition, we also investigate 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 [28], [30]. To find more special types of trinomial that can lead to better space and time complexity trade-off would be the future work.

APPENDIX A
PROOFS

A. Proof of Lemma 1

Proof The proof of this lemma mainly utilizes the reduction formulation (7). If the parameter $1 \leq \Delta \leq m-k-1$, 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 $-k \leq \Delta < 0$, then $0 < -\Delta \leq k$, 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. 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 $E_{s,t} \cdot x^{\delta_{s,t}}$ are in the range $[2\ell-2, 2\ell+2r-3]$, as $\deg(E_{s,t}) = 2\ell-2$ (for $0 \leq t < s < n-r$) or 2ℓ (for $0 < t < s, s \geq n-r$) 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. □

C. 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 $\ell \geq k$ 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 $p_2^{(i)} x^{\Delta_i - k}$ is given by:

$$\begin{aligned} p_2^{(i)} x^{\Delta_i - k} \bmod f(x) = &\sum_{j=0}^{m-1} h_j^{(i)} x^{-k+(j+\Delta_i) \bmod m} \\ &+ \sum_{j=m-\Delta_i}^{m-1} h_j^{(i)} x^{j+\Delta_i - m}, \end{aligned} \tag{23}$$

for $i = 1, 2, \cdots, \lambda$, $\lambda = \frac{n}{2}$ if n is even, and $\lambda = \frac{n-1}{2}$ if m 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 $i \neq j$. 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 (23), we immediately conclude the proposition.

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

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

But if $n \geq 5$, we have $(n-2)\ell \geq \frac{(n+1)\ell}{2} > k$. 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

$$\begin{aligned} S_2 x^{-2k} \bmod f(x) &= g_1 x^{\ell - 2k} \bmod f(x) \\ &= (p_1^{(1)} + p_1^{(1)} x^{-k} + p_2^{(1)} x^{-k}) x^{\ell - k}. \end{aligned}$$

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

$$\begin{aligned} p_2^{(1)} x^{\ell - 2k} \bmod f(x) = &\sum_{j=0}^{m-1} h_j^{(1)} x^{-k+(j+\ell-k) \bmod m} \\ &+ \sum_{j=0}^{k-\ell-1} h_j^{(1)} x^{j+\ell-k}, \end{aligned} \tag{24}$$

and

$$\begin{aligned} &p_1^{(1)} x^{\ell - 2k} \bmod f(x) \\ &= \left(p_{1,1}^{(1)} x^{k-\ell} + p_{1,2}^{(1)} \right) x^{\ell - 2k} \bmod f(x) \\ &= p_{1,1}^{(1)} x^{-k} + p_{1,2}^{(1)} (x^{m+\ell-2k} + x^{\ell-k}). \end{aligned} \tag{25}$$

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 (25), we have

$$\begin{aligned} &p_1^{(1)} x^{\ell-k} + p_{1,1}^{(1)} x^{-k} + p_{1,2}^{(1)} (x^{m+\ell-2k} + x^{\ell-k}) \\ &= p_{1,2}^{(1)} x^{m+\ell-2k} + p_{1,1}^{(1)} (1 + x^{-k}). \end{aligned} \tag{26}$$

We directly know that (26) and (24) 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 $p_2^{(i)} x^{\Delta_i - k}$ is the same as (14), while if $i = \frac{n-t}{2}, \cdots, \frac{n-1}{2}$, its modular reduction is the same as (15).

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:

$$\begin{aligned}
&\sum_{i=1}^{\frac{n-1}{2}} p_1^{(i)} x^{\Delta_i} + \sum_{i=\frac{n-t}{2}}^{\frac{n-1}{2}} p_1^{(i)} x^{\Delta_i} + p_{1,1}^{(\frac{n-t}{2})} x^{-k} + p_{1,2}^{(\frac{n-t}{2})} x^{t\ell-k} \\
&= \sum_{i=1}^{\frac{n-t}{2}-1} p_1^{(i)} x^{\Delta_i} + p_{1,1}^{(\frac{n-t}{2})} (1 + x^{-k}),
\end{aligned} \tag{27}$$

$$\sum_{i=\frac{n-t}{2}+1}^{\frac{n-1}{2}} p_1^{(i)} x^{m+\Delta_i-k} + \sum_{i=1}^{\frac{n-t}{2}-1} p_1^{(i)} x^{\Delta_i-k} + p_{1,2}^{(\frac{n-t}{2})} x^{m+t\ell-2k}, \tag{28}$$

Similar with Case 2, one can easily check that the subexpressions of (28) have no overlapped terms with each other. However, if $t = 1$, it is possible here $\ell + r - 2 - (k - t\ell) > k$ and $p_{1,1}^{(\frac{n-t}{2})}$ is overlapped with $p_{1,1}^{(\frac{n-t}{2})} x^{-k}$, which can not be concatenated together. Thus, adding (27) and (28) can be implemented as a sum of at most three independent subexpressions. But $\sum_{i=1}^{\frac{n-1}{2}} p_1^{(2)} x^{\Delta_i-k} \bmod f(x)$ consists of at most $2 \cdot \frac{n-1}{2} = n-1$ subexpressions. Meanwhile, some of these subexpressions have no overlapped term with each other. It totally requires $\lceil \log_2(\frac{n-1}{2} + 3 + \max\{\frac{n-t}{2} - 1, \frac{t+1}{2}\}) \rceil \leq \lceil \log(n+2) \rceil T_X$ delay in parallel. □

APPENDIX B
FORMULATIONS OF THE EXAMPLE

Here, we give the explicit formulations with respect to $p_1+p_0=\sum_{i=0}^{13}e_i'x^i$, $g_1=\sum_{i=0}^{13}h_i^{(1)}x^i$ and $g_2=\sum_{i=0}^{13}h_i^{(2)}x^i$ of , i.e.,

$$
\begin{cases}
h^{(0)}=0,\\
h_1^{(1)}=u_0^{(3,0)}v_0^{(3,0)},\\
h_2^{(1)}=u_1^{(3,0)}v_0^{(3,0)}+u_0^{(3,0)}v_1^{(3,0)},\\
h_3^{(1)}=u_0^{(3,0)}v_2^{(3,0)}+u_2^{(3,0)}v_0^{(3,0)}+u_1^{(3,0)}v_1^{(3,0)},\\
h_4^{(1)}=u_0^{(3,0)}v_3^{(3,0)}+u_3^{(3,0)}v_0^{(3,0)}+u_2^{(3,0)}v_1^{(3,0)},\\
\qquad +u_1^{(3,0)}v_2^{(3,0)}+u_0^{(3,1)}v_0^{(3,1)},\\
h_5^{(1)}=u_1^{(3,0)}v_3^{(3,0)}+u_3^{(3,0)}v_1^{(3,0)}+u_2^{(3,0)}v_2^{(3,0)}\\
\qquad +u_0^{(3,1)}v_1^{(3,1)}+u_1^{(3,1)}v_0^{(3,1)},\\
h_6^{(1)}=u_2^{(3,0)}v_3^{(3,0)}+u_3^{(3,0)}v_2^{(3,0)}+u_0^{(3,1)}v_2^{(3,1)}\\
\qquad +u_2^{(3,1)}v_0^{(3,1)}+u_1^{(3,1)}v_1^{(3,1)},\\
h_7^{(1)}=u_3^{(3,0)}v_3^{(3,0)}+u_0^{(3,1)}v_3^{(3,1)}+u_3^{(3,1)}v_0^{(3,1)}\\
\qquad +u_2^{(3,1)}v_1^{(3,1)}+u_1^{(3,1)}v_2^{(3,1)}+u_0^{(3,2)}v_0^{(3,2)},\\
h_8^{(1)}=u_3^{(3,1)}v_1^{(3,1)}+u_1^{(3,1)}v_3^{(3,1)}+u_2^{(3,1)}v_2^{(3,1)}\\
\qquad +u_0^{(3,2)}v_1^{(3,2)}+u_1^{(3,2)}v_0^{(3,2)},\\
h_9^{(1)}=u_3^{(3,1)}v_2^{(3,1)}+u_2^{(3,1)}v_3^{(3,1)}+u_2^{(3,2)}v_0^{(3,2)}\\
\qquad +u_0^{(3,2)}v_2^{(3,2)}+u_1^{(3,2)}v_1^{(3,2)},\\
h_{10}^{(1)}=u_3^{(3,1)}v_3^{(3,1)}+u_3^{(3,2)}v_0^{(3,2)}+u_0^{(3,2)}v_3^{(3,2)}\\
\qquad +u_1^{(3,2)}v_2^{(3,2)}+u_2^{(3,2)}v_1^{(3,2)},\\
h_{11}^{(1)}=u_3^{(3,2)}v_1^{(3,2)}+u_1^{(3,2)}v_3^{(3,2)}+u_2^{(3,2)}v_2^{(3,2)},\\
h_{12}^{(1)}=u_3^{(3,2)}v_2^{(3,2)}+u_2^{(3,2)}v_3^{(3,2)},\\
h_{13}^{(1)}=u_3^{(3,2)}v_3^{(3,2)},
\end{cases}
$$

$$
\begin{cases}
h_0^{(2)}=u_0^{(1,0)}v_0^{(1,0)},\\
h_1^{(2)}=u_1^{(1,0)}v_0^{(1,0)}+u_0^{(1,0)}v_1^{(1,0)},\\
h_2^{(2)}=u_0^{(1,0)}v_2^{(1,0)}+u_2^{(1,0)}v_0^{(1,0)}+u_1^{(1,0)}v_1^{(1,0)},\\
h_3^{(2)}=u_0^{(1,0)}v_3^{(1,0)}+u_3^{(1,0)}v_0^{(1,0)}+u_2^{(1,0)}v_1^{(1,0)}\\
\qquad +u_1^{(1,0)}v_2^{(1,0)}+u_0^{(2,0)}v_0^{(2,0)},\\
h_4^{(2)}=u_1^{(1,0)}v_3^{(1,0)}+u_3^{(1,0)}v_1^{(1,0)}+u_2^{(1,0)}v_2^{(1,0)}\\
\qquad +u_0^{(2,0)}v_1^{(2,0)}+u_1^{(2,0)}v_0^{(2,0)},\\
h_5^{(2)}=u_0^{(2,0)}v_2^{(2,0)}+u_2^{(2,0)}v_0^{(2,0)}+u_1^{(2,0)}v_1^{(2,0)},\\
h_6^{(2)}=u_0^{(2,0)}v_3^{(2,0)}+u_3^{(2,0)}v_0^{(2,0)}+u_2^{(2,0)}v_1^{(2,0)}\\
\qquad +u_1^{(2,0)}v_2^{(2,0)}+u_0^{(2,1)}v_0^{(2,1)},\\
h_7^{(2)}=u_3^{(2,0)}v_1^{(2,0)}+u_1^{(2,0)}v_3^{(2,0)}+u_2^{(2,0)}v_2^{(2,0)}\\
\qquad +u_0^{(2,1)}v_1^{(2,1)}+u_1^{(2,1)}v_0^{(2,1)},\\
h_8^{(2)}=u_3^{(2,0)}v_2^{(2,0)}+u_2^{(2,0)}v_3^{(2,0)}+u_2^{(2,1)}v_0^{(2,1)}\\
\qquad +u_0^{(2,1)}v_2^{(2,1)}+u_1^{(2,1)}v_1^{(2,1)},\\
h_9^{(2)}=u_3^{(2,0)}v_3^{(2,0)}+u_3^{(2,1)}v_0^{(2,1)}+u_0^{(2,1)}v_3^{(2,1)}\\
\qquad +u_1^{(2,1)}v_2^{(2,1)}+u_2^{(2,1)}v_1^{(2,1)},\\
h_{10}^{(2)}=u_3^{(2,1)}v_1^{(2,1)}+u_1^{(2,1)}v_3^{(2,1)}+u_2^{(2,1)}v_2^{(2,1)},\\
h_{11}^{(2)}=u_3^{(2,1)}v_2^{(2,1)}+u_2^{(2,1)}v_3^{(2,1)},\\
h_{12}^{(2)}=u_3^{(2,1)}v_3^{(2,1)},\\
h_{13}^{(2)}=0,
\end{cases}
$$

$$\begin{cases} e'_0 = a_0b_0 + a_{13}b_{11} + a_{12}b_{12} + a_{12}b_{13}, \\ e'_1 = a_1b_0 + a_0b_1 + a_{13}b_{12} + a_{12}b_{13}, \\ e'_2 = a_{13}b_{13} + a_2b_0 + a_1b_1 + a_0b_2, \\ e'_3 = a_3b_3 + a_2b_1 + a_1b_2, \\ e'_4 = a_4b_3 + a_3b_4 + a_2b_2, \\ e'_5 = a_5b_3 + a_4b_4 + a_3b_5, \\ e'_6 = a_6b_6 + a_5b_4 + a_4b_5, \\ e'_7 = a_7b_6 + a_6b_7 + a_5b_5, \\ e'_8 = a_8b_6 + a_7b_7 + a_6b_8, \\ e'_9 = a_9b_6 + a_8b_7 + a_7b_8 + a_6b_9, \\ e'_{10} = a_{10}b_{10} + a_9b_7 + a_8b_8 + a_7b_9, \\ e'_{11} = a_{11}b_{10} + a_{10}b_{11} + a_9b_8 + a_8b_9, \\ e'_{12} = a_{12}b_{10} + a_{11}b_{11} + a_{10}b_{12} + a_9b_9, \\ e'_{13} = a_{13}b_{10} + a_{12}b_{11} + a_{11}b_{12} + a_{10}b_{13}, \end{cases}$$

Obviously, in parallel implementation, e'_is can be obtained using $2T_X$, $h_i^{(2)}$ and $h_i^{(1)}$ can be obtained using $\lceil \log_2 5 \rceil T_X$ and $\lceil \log_2 6 \rceil T_X$, respectively. These delays meet proposition 3 and the complexity analysis in Section 4.1.

After obtaining the coefficients of $p_1 + p_0, g_1$ and g_2, we can perform the SPB modular reduction related to $E'(x)(x^{10} + x^6 + x^3 + 1)x^{-10}$, g_1x^{9-10} and g_2x^{3-10}. Please note that in this case, g_1, g_2 contain at most 12 nonzero terms, which do not need to split into two parts. More explicitly,

$$E'(x)x^{10-10} = \sum_{j=0}^{13} e'_j x^{(j+5) \bmod 14} + \sum_{j=0}^{4} e'_{j+9}x^j;$$

$$E'(x)x^{6-10} = \sum_{j=0}^{13} e'_j x^{(j+1) \bmod 14} + e'_{13};$$

$$E'(x)x^{3-10} = \sum_{j=0}^{13} e'_j x^{(j-2) \bmod 14} + e'_1x^{-1} + e'_0x^{-2};$$

$$E'(x)x^{0-10} = \sum_{j=0}^{13} e'_j x^{(j-5) \bmod 14} + \sum_{j=0}^{4} e'_j x^{j-5};$$

$$g_1x^{9-10} = \sum_{j=0}^{13} h_j^{(1)} x^{(j+4) \bmod 14} + \sum_{j=0}^{3} h_{j+9}^{(1)} x^j;$$

$$g_2x^{3-10} = \sum_{j=0}^{13} h_j^{(2)} x^{(j-2) \bmod 14} + h_0^{(2)}x^{-2} + h_1^{(2)}x^{-1};$$

where $h_0^{(1)} = 0, h_{13}^{(2)} = 0$. Therefore, it is clear that both S_1x^{-10} and S_2x^{-10} can be obtained in less than $\lceil \log_2 6 \rceil = 3T_X$ delay, which meets the delay assertions presented in 3.2, and Proposition 3.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (Grant no. 61402393, 61601396).

REFERENCES

[1] Rudolf Lidl and Harald Niederreiter. *Finite Fields*, Cambridge University Press, New York, NY, USA, 1996.
[2] A. Karatsuba and Yu. Ofman. "Multiplication of Multidigit Numbers on Automata," *Soviet Physics-Doklady (English translation)*, vol. 7, no. 7, pp. 595–596, 1963.
[3] J. Von Zur Gathen and J. Gerhard. 2003. *Modern Computer Algebra (2 ed.)*. Cambridge University Press, New York, NY, USA.
[4] P.L. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 362-369, March 2005. doi: 10.1109/TC.2005.4
[5] H. Fan, J. Sun, M. Gu, and K.-Y. Lam. "Overlap-free Karatsuba-Ofman polynomial multiplication algorithms," *Information Security, IET*, vol. 4, no. 1, pp. 8–14, March 2010.
[6] A. Weimerskirch, and C. Paar, "Generalizations of the Karatsuba Algorithm for Efficient Implementations," *Cryptology ePrint Archive, Report 2006/224*, http://eprint.iacr.org/
[7] H. Fan, M. Gu, J. Sun and K.Y. Lam, "Obtaining more Karatsuba-like formulae over the binary field," *IET Information Security*, vol. 6, no. 1, pp. 14-19, March 2012.
[8] Ku-Young Chang, Dowon Hong and Hyun-Sook Cho. "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.

[9] Young In Cho, Nam Su Chang, Chang Han Kim, Young-Ho Park and Seokhie Hong. "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.

[10] M. Elia, M. Leone and C. Visentin. "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.

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

[12] H. Fan and M.A. Hasan. "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, Dec 2006.

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

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

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

[16] Yin Li, Yiyang Chen. "New bit-parallel Montgomery multiplier for trinomials using squaring operation," *Integration, the VLSI Journal*, vol. 52, pp.142–155, January 2016.

[17] Y. Li, X. Ma, Y. Zhang and C. Qi, "Mastrovito Form of Non-recursive Karatsuba Multiplier for All Trinomials," *IEEE Trans. Comput.*, vol. 66, no.9, pp.1573–1584, Sept. 2017.

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

[19] H. Shen and Y. Jin. "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.

[20] B. Sunar and Ç.K. Koç, "Mastrovito multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 48,no. 5, pp. 522–527, 1999.

[21] A. Halbutogullari and Ç.K. Koç, "Mastrovito multiplier for general irreducible polynomials," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 503–518, May 2000.

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

[23] N. Petra, D. De Caro, and A.G.M. Strollo, "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.

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

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

[26] 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.

[27] X. Xie, G. Chen, Y. Li, "Novel bit-parallel multiplierfor $GF(2^m)$ defined by all-one polynomial using generalized Karatsuba algorithm", *Information Processing Letters*, Volume 114, Issue 3, pp.140–146, 2014.

[28] Y. Li, Y. Zhang, X. Guo and C. Qi, "N-Term Karatsuba Algorithm and Its Application to Multiplier Designs for Special Trinomials," *IEEE Access*, vol. 6, pp.43056–43069, Jul. 2018.

[29] Y. Li, Y. Zhang, and X. Guo, "Efficient Nonrecursive Bit-Parallel Karatsuba Multiplier for a Special Class of Trinomials," *VLSI Design*, vol. 2018, Article ID 9269157, 7 pages, 2018. https://doi.org/10.1155/2018/9269157.

[30] S. Park, K. Chang, D. Hong and C. Seo, "Low Space Complexity $GF(2^m)$ Multiplier for Trinomials Using n -Term Karatsuba Algorithm," in IEEE Access, vol. 7, pp. 27047-27064, 2019. doi: 10.1109/ACCESS.2019.2901242

[31] Haining Fan and M. A. Hasan, "Relationship between $GF(2^m)$ Montgomery and Shifted Polynomial Basis Multiplication Algorithms," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1202-1206, Sept. 2006.