On the Complexity of non-recursive n-term Karatsuba Multiplier for Trinomials

Yin Li, Yu Zhang, Xingpo Ma and Chuanda Qi

Abstract

In this paper, we continue the study of bit-parallel multiplier using a n-term Karatsuba algorithm (KA), recently introduced by Li et al. (IEEE Access 2018). Such a n-term KA is a generalization of the classic KA, which can multiply two n-term polynomials using $O(n^2/2)$ scalar multiplications. Based on this observation, Li et al. developed an efficient bit-parallel multiplier scheme for a new special class of irreducible trinomial $x^m + x^k + 1, m = nk$. The lower bound of the space complexity of their proposal is about $O(\frac{m^2}{2} + m^{3/2})$. However, such a special type of trinomial does not always exist. In this contribution, we investigate the space and time complexity of Karatsuba multiplier for general trinomials, i.e., $x^m + x^k + 1$ where $m > 2k$. We use a new decomposition that $m = n\ell + r$, where $r < n, r < \ell$. Combined with shifted polynomial basis (SPB), a new approach other than Mastrovito approach is proposed to exploit the spatial correlation between different subexpressions. Explicit space and time complexity formulations are given to indicate the optimal choice of the decomposition. As a result, the optimal multiplier achieves nearly the same space complexity as $x^m + x^k + 1, m = nk$, but it is suitable to more general trinomials. Meanwhile, its time complexity matches or is at most $1T_X$ higher than the similar KA multipliers, where T_X is the delay of one 2-input XOR gate.

Index Terms

Bit-parallel multiplier, n-Karatsuba algorithm, shifted polynomial basis, optimal, trinomials.

I. Introduction

There has been 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 by $GF(2^m)$, field multiplication is one of the most frequently desired operations, as other complex operations such as exponentiation and division can be implemented by iterative multiplication. Therefore, it is critical to design a suitable $GF(2^m)$ multiplier under conditions of the different hardware resources.

Generally speaking, there are three type of bit-parallel multipliers of different architecture, i.e., quadratic [16], [17], [25], [26], subquadratic [6], [5] or hybrid bit-parallel multipliers [14], [12], [24], [13], [18]. Quadratic multiplier normally utilize schoolbook approach to implement the polynomial multiplication (polynomial basis in $GF(2^m)$), while subquadratic or hybrid methods usually apply 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 usually smaller than other two types of multipliers. Nevertheless, their time complexities are often larger than quadratic or hybrid counterparts. Conversely, the hybrid multipliers can provide a trade-off between the time and space complexities [14]. Some of these approaches can save about 1/4 logic gates compared with the quadratic multipliers, while the time complexity cost only one more T_X compared with the fastest quadratic multipliers [21], [33] In these schemes, the KA is frequently applied to compute the product of two degree polynomials.

The Karatsuba algorithm is a classic divide and conquer algorithm, which optimize polynomial multiplication by partitioning each polynomial into two halves and utilizing three sub-multiplications instead of four ones. Besides the classic KA, there are several variations, e.g. generalized n-term KA introduced by Weimerskirch and Paar [6] and 4, 5 and 6-term of KA introduced by [4]. The former algorithm split each polynomial into n terms and apply KA strategy for every two sub-polynomial. The latter one introduced new formulae to minimize the number of sub-multiplications. Based on Montgomery's work, various combinations of these formulae resulted in remarkable improvements for higher degree polynomial multiplications [7]. However, these Karatsuba-like formulae usually contains complicated linear combinations of the split parts, which will lead more gates delay for bit-parallel multiplier. Conversely, Weimerskirch and Paar's approach is more fit for constructing hybrid multipliers [34], [33]. We call this algorithm as n-term KA and use this notion thereafter.

Recently, Li et al. investigate the application of n-term KA ($n \geq 3$) to a special class of trinomial $x^m + x^k + 1, m = nk$ [33]. However, this type of irreducible trinomials is not abundant, so that put a confinement to the application of these schemes. Inspired by this point, in this regards, we focus on an extension of this proposal. 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, we assume that $m = n\ell + r$ with $r < n, r < \ell$. We also use shifted polynomial basis (SPB) to optimize the reduction. The main architecture is described in details. In addition, explicit formulae for space and time complexity are given to indicate the optimal selection of n and ℓ. As

Yin Li, Yu Zhang, Xingpo Ma and Chuanda Qi are with Department of Computer Science and Technology, Xinyang Normal University, Henan, P.R.China, 464000. email: yunfeiyangli@gmail.com (Yin Li).

a main contribution, we show that the lower bound of our proposal is $O(\frac{m^2}{2}+\frac{m^{3/2}}{4})$, which matches the optimal result of [33].

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. A new bit-parallel multiplier architecture is then proposed in Section 3. Section 4 presents an analysis of our proposal and study the optimal parameters of KA. 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 [15] 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 $M=\{x^{m-1},\cdots,x,1\}$ constitutes a polynomial basis (PB). Then, the SPB can be obtained by multiplying the set M by a certain exponentiation of x:

Definition 1 *[15] 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:

$$C(x)x^{-v}=A(x)x^{-v}\cdot B(x)x^{-v} \bmod f(x).$$

If the parameter v is properly selected, the polynomial modular reduction using SPB representation is simpler than that using PB representation, especially for irreducible trinomial or some type of pentanomials [16]. Specially, for trinomial x^m+x^k+1, it has been proved that the optimal value of v is k or $k-1$ [15], [16]. In this paper, we choose that $v=k$ and use this denotation thereafter.

B. n-term Karatsuba Algorithm

The classic KA multiply two 2-term polynomials using three scalar multiplications at the cost of one extra addition. Then, Weimerskirch and Paar [6] gave a generalized formulae that is applicable for the polynomial multiplication with higher degree. We denote such an algorithm as n-term KA ($n\ge 2$). Firstly, assume that there are two n-term polynomial with $n-1$ degree over $\mathbb{F}_2$:

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

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

$$D_i=a_ib_i.$$

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

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

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

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

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

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

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. Besides this algorithm, Montgomery [4] and Fan [7] proposed more Karatsuba-like formulae. These formulae aim to decrease as many coefficient multiplications as possible. While their formulae contain more additions that require more gate delay for parallel implementation.

In the following section, we will investigate 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

We firstly investigate the multiplication of two polynomials of degree $m-1$ using n-term KA. Then, the modular reduction for related results are considered. Accordingly, we propose an efficient bit-parallel n-term Karatsuba multiplier architecture. The space and time complexity of corresponding multiplier is also discussed.

Provide that $f(x)=x^m+x^k+1$ be an irreducible trinomial that define 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,

$$A=\sum_{i=0}^{m-1}a_ix^i,\ B=\sum_{i=0}^{m-1}b_ix^i.$$

The 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.,

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

A. Polynomial multiplication using n-term Karatsuba algorithm

Now we partition $A=\sum_{i=0}^{m-1}a_ix^i$ and $B=\sum_{i=0}^{m-1}b_ix^i$ into n parts and apply n-term KA to the polynomial multiplication. However, m is not always divisible by n. We assume that $m=n\ell+r$, 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 ones consisting of $\ell+1$ bits. More explicitly,

$$\begin{aligned}
A=&A_{n-1}x^{(n-1)\ell+r-1}+\cdots+A_{n-r+1}x^{(n-r+1)\ell+1}+A_{n-r}x^{(n-r)\ell}\\
&+A_{n-r-1}x^{(n-r-1)\ell}+\cdots+A_1x^{\ell}+A_0,
\end{aligned}$$

and

$$\begin{aligned}
B=&B_{n-1}x^{(n-1)\ell+r-1}+\cdots+B_{n-r+1}x^{(n-r+1)\ell+1}+B_{n-r}x^{(n-r)\ell}\\
&+B_{n-r-1}x^{(n-r-1)\ell}+\cdots+B_1x^{\ell}+B_0,
\end{aligned}$$

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:*

$$\begin{aligned}
AB=&\Big(A_{n-1}B_{n-1}x^{(n-1)\ell+r-1}+A_{n-2}B_{n-2}x^{(n-2)\ell+r-2}+\\
&\cdots+A_{n-r}B_{n-r}x^{(n-r)\ell}+\cdots+A_1B_1x^{\ell}+A_0B_0\Big)\cdot h(x)\\
&+\sum_{i=1}^{2n-3}\Big(\sum_{\substack{s+t=i,\\ n>s>t\geq 0}}D_{s,t}\Big)x^{i\ell+\delta_{s,t}},
\end{aligned}\tag{2}$$

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

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

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

$$\begin{aligned}A &= A_{n-1}x^{\overline{n-1}} + A_{n-2}x^{\overline{n-2}} + \cdots + A_1 x^{\overline{1}} + A_0 x^{\overline{0}},\\ B &= B_{n-1}x^{\overline{n-1}} + B_{n-2}x^{\overline{n-2}} + \cdots + B_1 x^{\overline{1}} + B_0 x^{\overline{0}},\end{aligned}$$

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

$$\begin{aligned}AB &= \sum_{i=0}^{n-1} A_i B_{n-1} x^{\overline{i}+\overline{n-1}} + \cdots + \sum_{i=0}^{n-1} A_i B_0 x^{\overline{i}+\overline{0}}\\ &= \sum_{i=0}^{n-1} A_i B_i x^{2\overline{i}} + \sum_{0\le i<j<n} (A_i B_j + A_j B_i) x^{\overline{i}+\overline{j}}\end{aligned} \tag{4}$$

Applying (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, (4) can be rewritten as:

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

When we substitute the symbol $\overline{i}$ with original degree, it is clear that we get above expression. $\square$

Analogous to the approach present in [21], we can divide (2) into two parts and compute them independently, i.e.,

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

Therefore, the SPB field multiplication is given by

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

In order to obtain the final result, we have to perform the modular reduction with respect to S_1x^{-2k} and S_2x^{-2k}. In the following section, we study these modular reduction, respectively.

B. Computation of $S_1x^{-2k} \bmod f(x)$

Note that S_1 is written as a product of two expressions. Let

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

We first consider the computation of $E(x)$ and then investigate the modular reduction of S_1x^{-2k} modulo $x^m + x^k + 1$.

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 $(n-1)\ell + r - 1 + 2\ell = m + \ell - 1$. Its coefficients are given by

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

The modular reduction with respect to $E(x)$ 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 $d(x)$ denote $p_1 + p_0$. The coefficients of $d(x)$ can be obtained by adding the ℓ most significant bits of $E(x)$ to its ℓ least significant bits, i.e.,

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

where d_is represent the coefficients of $d(x)$.

We then consider the modular reduction of $S_1 x^{-2k}$. Note that

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

One one hand, since p_1 consist of ℓ terms and $h(x) = x^{(n-1)\ell+r-1} + x^{(n-2)\ell+r-2} + \cdots + x^{\ell} + 1$, there is no overlapped term between $p_1 x^{i\ell+\epsilon_i}$ and $p_1 x^{(i-1)\ell+\epsilon_{i-1}}$, for $i = 1, 2, \cdots, n-1$. Specifically, the symbol ϵ_i represent the extra term degree in $h(x)$, where $\epsilon_i = i-n+r$ if $\epsilon_i > 0$ and 0 otherwise. Also, one can check that $\deg(p_1 h x^{-k}) = (n-1)\ell + r - 1 + \ell - 1 - k = m - k - 2$, and all the term degree of $p_1(x)h(x)x^{-k}$ are in the range $[-k, m-k-1]$. Therefore, $p_1(x)h(x)x^{-k} \bmod f(x) = p_1(x)h(x)x^{-k}$ and no XOR gate is needed to compute this expression.

On the other hand, notice that $d(x)h(x)x^{-2k} = \sum_{i=0}^{n-1} d(x)x^{-k} \cdot x^{i\ell+\epsilon_i-k}$. Because $d(x)$ is of the degree $m-1$, $d(x)x^{-k}$ can be seen as an element of $GF(2^m)$ in SPB representation. The modular reduction of $d(x)x^{-k} \cdot x^{i\ell+\epsilon_i-k}$ modulo $f(x)$ is shifting $d(x)x^{-k}$ by $i\ell + \epsilon_i - k$ bits in such a finite field. These operations depend on the magnitude relations between k and ℓ. Recall that $k \le m/2$, and $m = n\ell + r, n > r, \ell > r$. Two cases are considered:

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

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

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

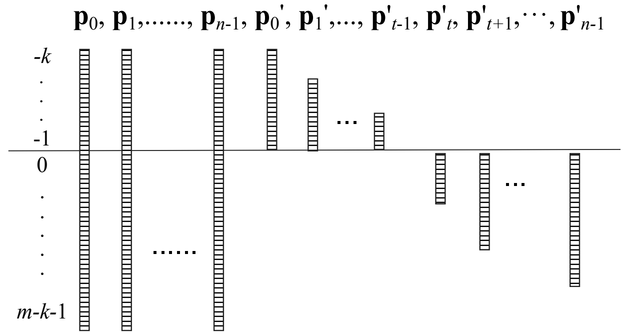

Fig. 1. Bit positions for all the subexpressions.

Therefore, the case of $k \geq (n-1)\ell$ happens only if $n = 2$. Related explicit modular reduction has already been studied in [21], thus, we only analyze the case of $k < (n-1)\ell$ in this study. The reduction relies 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*

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

The proof of above lemma can be found in the appendix. Lemma 1 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 this lemma, we can perform the modular reduction with respect to $d(x)x^{-k} \cdot x^{i\ell+\epsilon_i-k}$. Please notice that $i\ell + \epsilon_i - k$ is equivalent to 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 $d(x)x^{i\ell+\epsilon_i-2k} \bmod f(x)$ are given by:

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

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

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

for $i = t, \cdots, n-1$, where $\theta_i = i\ell + \epsilon_i - k$.

One can easily check that each of these expressions in (8) and (9) consists 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 $\sum_{j=0}^{-\theta_i-1} d_j x^{j+\theta_i}$ ($i = 0, 1, \cdots, t-1$) have all their terms degrees smaller than 0, while $\sum_{j=m-\theta_i}^{m-1} d_j x^{j+\theta_i-m}$ ($i = t, \cdots, n-1$) have all their terms degrees larger than 0. That is to say, there is no overlapped terms between these two kinds of subexpressions. We can add them without any logic gates. The Figure 1 depicts bit positions for these subexpressions.

The vectors $\mathbf{P}_i, \mathbf{P}'_i$ in Figure 1 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 need to be added. In parallel implementation, it only need $\lceil \log_2(n+1+\max\{t, n-t\}) \rceil T_X$

to add all these subexpressions together using 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, after we finish computing p_1+p_0 and p_1.

C. Computation of $S_2x^{-2k} \bmod f(x)$

The computation of S_2x^{-2k} modulo $f(x)$ is different from that of S_1x^{-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 know that A_i, B_i for $i = 0, 1, \cdots, n-r-1$ have the degrees $\ell-1$ and the rest of A_i, B_i have degrees ℓ. Thus, 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 $0 \leq t < s, s \geq n-r$. However, in order to perform modular reduction for S_2x^{-2k}, we can apply a trick established in [32] to categorize all the $D_{s,t}$s, where the $D_{s,t}$s from the same category can be recognized as a integral to perform modular reduction. We have the following proposition.

Proposition 2 *S_2 can be expressed as the plus of $g_1x^{(2\lambda-1)\ell}$, $g_2x^{(2\lambda-3)\ell}, \cdots, g_\lambda x^\ell$ 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 $n > s > t \geq 0$.

Proof The proof about this proposition can be built using mathematical induction. Please see Section 3.2 in [32]. □

Therefore, based on Proposition 2,

$$S_2x^{-2k} = g_1x^{(2\lambda-1)\ell-2k} + g_2x^{(2\lambda-3)\ell-2k} + \cdots + g_\lambda x^{\ell-2k}.$$

Accordingly, its modular reduction by $f(x)$ can also be expressed as a plus of these λ subexpressions modulo $f(x)$. We can perform these modular reductions in parallel and then add the results together. The detailed computation for $S_2x^{-2k} \bmod f(x)$ is presented as follows:

(i) Perform bitwise addition $A_s + A_t, B_s + B_t, (n > s > t \geq 0)$ in parallel.
(ii) Classify the subexpressions $D_{s,t}$ into λ parts according to Proposition 2 and constitute these $D_{s,t}$ to λ bigger ones, i.e., $g_1, g_2, \cdots, g_\lambda$.
(iii) Perform reductions of $g_1x^{(2\lambda-1)\ell-2k}, g_2x^{(2\lambda-3)\ell-2k}, \cdots, g_\lambda x^{\ell-2k}$ modulo $f(x)$.
(iv) Add all these results binary XOR tree to obtain the $S_2x^{-2k} \bmod f(x)$.

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 cost one T_X delay. Meanwhile, one can easily check that the classification in Step (ii) does not cost any logic gates, but this step also includes the polynomial multiplications related to $g_1, \cdots, g_\lambda$. These computations are analogous to that of $E(x)$ in Section 3.2. The reduction of S_2x^{-2k} is performed in Step (iii) and Step (iv). Note that these steps can be computed jointly.

As the 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 $D_{s,t}$. 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 $(n-1)\ell+2\ell+2r-3=m+\ell+r-3$. Provide 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 be also 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 $0 \le t < s < n-r$. 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 $D_{s,t}x^{\delta_{s,t}}$s 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 $s_1 \ge s_2 \ge s_3$ and $s_1+t_1=s_2+t_2+1=s_2+t_2+2$.

From (3), it is easy to obtain that $\delta_{s_1,t_1} \ge \delta_{s_2,t_2} \ge \delta_{s_3,t_3}$. One can easily check that only if $\delta_{s_1,t_1}=\delta_{s_2,t_2}=\delta_{s_3,t_3}$, corresponding coefficients of g_i are overlapped by these three subexpressions. Part of its coefficients are given by:

$$h_j^{(i)} = \begin{cases} \vdots & \vdots \\ d^{(s_3,t_3)}_{j-(s-3)\ell-\delta} + d^{(s_2,t_2)}_{j-(s-2)\ell-\delta}, & (s-2)\ell+\delta \le j \le \\ & \quad (s-1)\ell+\delta-1, \\ d^{(s_3,t_3)}_{2\ell} + d^{(s_2,t_2)}_{\ell} + d^{(s_1,t_1)}_{0}, & j=(s-1)\ell+\delta, \\ d^{(s_2,t_2)}_{j-(s-2)\ell-\delta} + d^{(s_1,t_1)}_{j-(s-1)\ell-\delta}, & (s-1)\ell+\delta+1 \le j \\ & \quad \le s\ell+\delta-1, \\ \vdots & \vdots \end{cases}$$

where $\delta=\delta_{s_1,t_1}=\delta_{s_2,t_2}=\delta_{s_3,t_3}$. We note that in this case, $h^{(i)}_{(s-1)\ell+\delta}$ 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 $h^{(i)}_{(s-1)\ell+\delta}$ contains $\ell+3$ terms of $u_i^{s,t} \cdot v_i^{s,t}$, which lead to at most $\lceil \log_2(\ell+3) \rceil T_X$ delay using binary XOR tree. Also notice that one T_A is need 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 computation of Step (iii) and (iv). Firstly we have a following observation.

Observation 3.3.1 The modular reduction of $g_1 x^{(2\lambda-1)\ell-2k}$, $g_2 x^{(2\lambda-3)\ell-2k}$, $\cdots, g_\lambda x^{\ell-2k}$ by $f(x)$ only require one reduction step.

The proof of this observation is given in 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 expressions $g_1 x^{(2\lambda-1)\ell-2k}, g_2 x^{(2\lambda-3)\ell-2k}, \cdots, g_\lambda x^{\ell-2k}$ can be rewritten in a unified form, i.e.,

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

Please notice that the explicit reduction formulations of $g_i x^{\Delta_i-k}$ modulo $f(x)$ depend on the choice of n, ℓ and k. According to previous statement, it is clear that $n \ge 2$ and thus $\ell \le m/2$. We also have $0<k\le m/2$. 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 $\ell-2k>-k$. Thus, all the terms of $g_i x^{\Delta_i-k}$ have their degrees larger than $-k$. We only need to reduce the terms whose degrees are greater than $m-k-1$. Therefore, to investigate the modular reduction details, six cases are considered:

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

As described in Section 3.2, Case 1 happens only if $n=2$, which has already been studied in [21], 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.,

$$g_i = p_1^{(i)} x^m + p_2^{(i)}, \tag{12}$$

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

$$g_i \bmod f(x) = p_1^{(i)}(x^k+1)+p_2^{(i)}.$$

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

$$g_i x^{\Delta_i - k} \bmod f(x) = \Big(p_1^{(i)} + p_1^{(i)} x^{-k} + p_2^{(i)} x^{-k}\Big) x^{\Delta_i} \bmod f(x), \tag{13}$$

$i = 1, 2, \cdots, \lambda$. Consider the term degree range of SPB representation, the expressions $p_1^{(i)}$, $p_1^{(i)} x^{-k}$ and $p_2^{(i)} x^{-k}$ have all their term degrees in the range $[-k, m-k-1]$. Therefore, the modular reductions of $g_i x^{\Delta_i - k}$ will also utilize Lemma 1. We then have following proposition.

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

Proof Obviously, Step (iii) and (iv) actually compute $\sum_{i=1}^{\lambda} g_i x^{(2\lambda - 2i+1)\ell - 2k} \bmod f(x)$, which consists of polynomial modular reductions and additions. Without loss of generality, we only analyze Case 2, 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:

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

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, we consider the reduction of $p_1^{(i)} x^{\Delta_i} + p_1^{(i)} x^{\Delta_i - k}$. We know that the max 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 $p_1^{(i)} x^{\Delta_i}$ 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 the term degree 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 $i = \frac{n-t+3}{2}, \cdots, \frac{n}{2}$. Meanwhile,

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

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].

[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.

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 is no overlapped terms among $p_1^{(1)}x^{\Delta_1}, p_1^{(2)}x^{\Delta_2}, \cdots, p_1^{(\frac{n}{2})}x^{\Delta_{\frac{n}{2}}}$, 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 subexpressions 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 subexpressions that has no overlapped terms.

Finally, we add all the modular reduction results included in (14), (15), (19) and (20) to obtain $S_2x^{-2k} \bmod f(x)$. 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 1. 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. $\square$

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 use 4-term Karatsuba algorithm to the polynomial multiplication. It is clear that $14 = 4\times 3+2$. We have $n = 4, \ell = 3, r = 2$ and r satisfies $r < n, r < \ell$. 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

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

According to equation (2), then

$$\begin{aligned}
A \cdot B &= (A_3B_3x^{10} + A_2B_2x^6 + A_1B_1x^3 + A_0B_0)h(x) + D_{3,2}x^{16}\\
&\quad + D_{3,1}x^{13} + D_{3,0}x^{10} + D_{2,1}x^9 + D_{2,0}x^6 + D_{1,0}x^3\\
&= S_1 + S_2,
\end{aligned}$$

where $h(x) = x^{10} + x^6 + x^3 + 1$, $D_{s,t} = \sum_{i=0}^{6} d_i^{(s,t)}x^i$ for $3 \geq s > 1, 2 \geq t > 0, s \neq t$ and $D_{1,0} = \sum_{i=0}^{4} d_i^{(1,0)}x^i$. Apparently, there are $\binom{5}{2} = 10$ such $D_{s,t}$s.

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 $g_1 = D_{3,2}x^{6+1} + D_{3,1}x^{3+1} + D_{3,0}x$, $g_2 = D_{2,1}x^6 + D_{2,0}x^3 + D_{1,0}$. Clearly, $\delta_{3,2} = \delta_{3,1} = \delta_{3,0} = 1$ and the rest of $\delta_{s,t}$s are all zero.

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

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

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,

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

It is easy to check that $p_1(x^{10} + x^6 + x^3 + 1)x^{-5}$ has all its terms in the range $[-5, 8]$ does not need any logic gates. We also can easily obtain the reduction of $d(x)h(x)x^{-10}$ and g_1x^{9-10}, g_2x^{-10} modulo $x^{14} + x^5 + 1$.

IV. Complexity Analysis

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

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

As presented in section 3.2, the computation of S_1x^{-2k} modulo $f(x)$ consists of computation of $p_1, p_1 + p_0$ following a modular multiplication by $h(x)x^{-2k}$. We first investigate the complexity of p_1 and $d(x) = p_1 + p_0$. 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 d_i are composed of at most $\ell + 1$ coefficient products of $A_iB_i, i = 0, 1, \cdots, n-1$. We immediately conclude that $p_1 + p_0$ and p_1 can be computed in $T_A + \lceil \log_2(\ell + 1) \rceil T_X$ delay. Table I presents the gate count and time delay for implementation of each coefficient of $p_1 + p_0$.

Furthermore, notice that

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

$p_1 + p_2$ contains all the terms that 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 [20], [21]. The authors found that if two binary XOR trees share k common items, only $k - W(k)$ XOR gates can be saved, where $W(k)$ is the Hamming weight of the binary representation of k. We can easily check that the coefficients of p_1 shares $1, 2, \cdots, \ell$ items with $p_1 + p_0$, which requires $\sum_{i=1}^{\ell} W(i) - \ell$ XOR gates in all.

We then investigate the complexity of $d(x)h(x)x^{-2k} + p_1h(x)x^{-k}$. As shown in Section 3.2, we only need to add $2n + 1$ expressions to obtain the result. Please notice that some of the expressions in (8) and (9) can be combined together. More explicitly, 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 $S_1 \bmod f(x)$.

TABLE I
THE COMPUTATION COMPLEXITY OF d_i

d_i	#AND	#XOR	Delay
$d_0 = c_0^{(0)} + c_{\ell+1}^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil)T_X$
$d_1 = c_1^{(0)} + c_{\ell+2}^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil)T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$d_{\ell-1} = c_{\ell-1}^{(0)} + c_{2\ell}^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil)T_X$
$d_\ell = c_\ell^{(0)} + c_0^{(1)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil)T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$d_{2\ell-1} = c_{\ell-1}^{(1)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil)T_X$
$d_{2\ell} = c_\ell^{(1)} + c_0^{(2)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil)T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$d_{(n-r+1)\ell-1} = c_{\ell-1}^{(n-r)}$	ℓ	$\ell-1$	$T_A + (\lceil \log_2 \ell \rceil)T_X$
$d_{(n-r+1)\ell} = c_\ell^{(n-r)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil)T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$d_{(n-1)\ell+r-1} = c_{\ell+1}^{(n-2)} + c_0^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil)T_X$
$d_{m-1} = c_\ell^{(n-1)}$	$\ell+1$	ℓ	$T_A + (\lceil \log_2(\ell+1) \rceil)T_X$
Total	$(n-r)\ell^2 + r(\ell+1)^2$	$(n-r)\ell(\ell-1) + r\ell(\ell+1)$	$T_A + (\lceil \log_2(\ell+1) \rceil)T_X$

TABLE II
SPACE AND TIME COMPLEXITIES OF $S_1 \bmod f(x)$

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

B. Space and time complexity of S_2

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

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

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 $D_{s,t}$ cost 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, which requires $m + r - 3 - 2\ell$ XOR gates (even n) or $m + r - 3 - \ell$ 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 explicit space and time complexities for each steps are summarized in Table 3.

C. Theoretic Complexity

As mentioned in previous section, $S_1 x^{-2k} \bmod f(x)$ and $S_2 x^{-2k} \bmod f(x)$ are computed in parallel and the overall circuit delay is equal to the longer delay of either $S_1 x^{-2k} \bmod f(x)$ or $S_2 x^{-2k} \bmod f(x)$. From Table 2 and 3, it is clear

TABLE III
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 $D_{s,t}$	-	$\frac{n}{2}(m + r - 3 - 2\ell)$ (even n) $\frac{n-1}{2}(m + r - 3 - \ell)$ (odd n)	
(iii), (iv)	Case 2	-	$\frac{mn}{2} + (n - \frac{t-1}{2})(r-3) + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert$	$\lceil \log_2(\frac{n+4}{2} + \max\{\frac{n-t-1}{2}, \frac{t+1}{2}\}) \rceil T_X$
	Case 3	-	$\frac{mn}{2} + n(r-3) + \sum_{i=1}^{n/2} \lvert\Delta_i\rvert$	$\lceil \log_2(n+2) \rceil T_X$
	Case 4 ($n=3$)	-	$m + 2\ell + 2r - 6$	$\lceil \log_2 5 \rceil T_X$
	Case 5	-	$\frac{(n-1)m}{2} + (n - \frac{t+1}{2})(r + \ell - 3) + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert$	$\lceil \log_2(\frac{n+5}{2} + \max\{\frac{n-t-2}{2}, \frac{t+1}{2}\}) \rceil T_X$
	Case 6	-	$\frac{(n-1)m}{2} + (n-1)(r + \ell - 3) + \sum_{i=1}^{(n-1)/2} \lvert\Delta_i\rvert$	$\lceil \log_2(n+1) \rceil T_X$
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 $t\ell \leq k, (t+2)\ell > k$				

that the circuit delay of $S_2x^{-2k} \bmod f(x)$ is slightly higher. Thus the overall circuit delay for parallel implementation of S_1x^{-2k}, S_2x^{-2k} modulo $f(x)$ is $T_A + (1 + \lceil \log_2(\ell+3) \rceil + \lceil \log_2(n+2) \rceil)T_X$. Afterwards, m more XOR gates are needed to add these two results, which lead to one more T_X delay. To sum up, the 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 [2]

$$\begin{aligned}
&\text{\# AND: } \frac{m^2}{2} + \frac{m\ell}{2} + (m + n + \frac{\ell+1}{2})r - (\ell+2)r^2, \\
&\text{\# XOR: } \frac{m^2}{2} + (2n + \frac{\ell}{2} + r - 1)m + \frac{n^2 + rn + r + \ell r}{2} + \sum_{i=1}^{\ell} W(i) \\
&+ \sum_{i=0}^{n-1} |\theta_i| + \sum_{i=1}^{n/2} |\Delta_i| - \ell r^2 - \ell - 5n, (n \text{ even}), \\
&\text{or} \\
&\frac{m^2}{2} + (2n + r + \frac{\ell-1}{2})m + \frac{n^2 + rn + \ell r + 9}{2} + \sum_{i=1}^{\ell} W(i) \\
&+ \sum_{i=0}^{n-1} |\theta_i| + \sum_{i=1}^{(n-1)/2} |\Delta_i| - \ell r^2 - 5n - \frac{5r + 3\ell}{2}, (n \text{ odd}),
\end{aligned}$$

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 parameters (i.e., n, ℓ, r) we choose, the corresponding multiplier requires at least $m^2/2$ AND gates as well as $m^2/2$ 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 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 larger than $\frac{5m^2}{2}$.

Specifically, as r is a small integer, the functions related to r can roughly be recognized as a linear function of m. Thus, the space complexity of our proposal depends on the parameter n, ℓ, m. Note that $\sum_{i=0}^{\ell} W(i)$ can be roughly written as $\frac{\ell}{2} \log_2 \ell$ [20]. Therefore, if we ignore these linear or small parts, 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 influence the space and time complexity of our proposal a lot. As we presented in Figure 1, the time delay of adding these vectors $\mathbf{P}_i, \mathbf{P}'_i$ in parallel is $\lceil \log_2(n + 1 + \max\{t, n-t\}) \rceil$, where t satisfies

$$(t-1)\ell + \epsilon_{t-1} \leq k < t\ell + \epsilon_t.$$

[2]For simplicity, we omit certain small number presented in Table 2-3.

It is obvious that when t approaches $n/2$, we obtain the minimal time delay. We then directly obtain that k is close to $(n/2)\cdot\ell \approx m/2$. Meanwhile, from the proof of Proposition 3, the computation 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,

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

and

$$\begin{aligned}\sum_{i=1}^{\lambda}|\Delta_i| = &\frac{(t'+1)k}{2} + \sum_{i=(t'+3)/2}^{\lambda}(2i-1)\ell - \sum_{i=1}^{(t'+1)/2}(2i-1)\ell\\ &-(\lambda - \frac{t'+1}{2})k\end{aligned}$$

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

In order to inspect the variation tendency of above expressions, we omit the small parameter ϵ_i and construct to functions with respect to t and t'.

$$\begin{aligned}f_1(t) &= (2t-n)k + (-t^2 + t + \frac{n^2-n}{2})\ell,\\ f_2(t') &= (t'+1-\lambda)k + \frac{(-t'^2 - 2t' + 2\lambda^2 - 1)\ell}{2}.\end{aligned}$$

We can roughly know that the bigger of the parameters t and t', the smaller of two functions. That is to say, bigger k can lead to a lower space complexity. To sum up, trinomial $x^m + x^k + 1, m \ge 2k$ with bigger k is more suitable to develop hybrid Karatsuba multiplier. In fact, [24] 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*:* If k is fixed, the choice of n, ℓ can determine the space complexity of our proposal. From previous description, we know that k highly influence the values of $\sum_{i=0}^{n-1}|\theta_i|$ and $\sum_{i=1}^{(n-1)/2}|\Delta_i|$. If $k=1$, then $t=1, t'=0$. These subexpressions reaches their maximum value, i.e., $\max\sum_{i=0}^{n-1}|\theta_i| = \frac{n(n-1)\ell}{2} + \frac{r(r-1)}{2}$,$\max\sum_{i=1}^{(n-1)/2}|\Delta_i| = \frac{(2\lambda^2-1)\ell}{2} + 1 - \lambda$ ($\lambda = \frac{n}{2}$ or $\frac{n-1}{2}$). All these subexpression now have the values of $O(n^2\ell)$. 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:

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

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

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

The optimal n, ℓ are varies according to k.

V. More discussion

As shown in previous sections, the time delay of our proposal is less than $T_A + (2 + \lceil\log_2(\ell+3)\rceil + \lceil\log_2(n+2)\rceil)T_X$. For some special type of trinomials, this delay can be improved further. In [], we have shown that for $x^m + x^k + 1, m = nk$, a speedup strategy can apply to decreased the time complexity to $T_A + (\lceil\log_2 k\rceil + \lceil\log_2 3n\rceil)T_X$. However, the precondition to apply such a speedup strategy is that delay of $S_1 \bmod f(x)$ is lower than that of $S_2 \bmod f(x)$ by a least one T_X. If these delays are equal, no speedup can achieve. To find more types of trinomials that can apply this speedup strategy is our future work.

In Table 4, 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 other hybrid multiplier. The best of our result only costs about $O(\frac{m^2}{2} + \frac{m^{3/2}}{4})$ circuit gates compared with the previous architectures (quadratic or hybrid). On the other hand, the time complexity of the proposed multiplier is very closed to the fastest result utilizing classic Karatsuba algorithm.

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

Multiplier	# AND	# XOR	Time delay
Montgomery[30], school-book[29]	m^2	m^2-1	$T_A+(2+\lceil\log_2 m\rceil)T_X$
Mastrovito [25][26][27]	m^2	m^2-1	$T_A+(2+\lceil\log_2 m\rceil)T_X$
Mastrovito [28]	m^2	m^2-1	$T_A+(\lceil\log_2(2m+2k-3)\rceil)T_X$
SPB Mastrovito [16]	m^2	m^2-1	$T_A+\lceil\log_2(2m-k-1)\rceil T_X$
Montgomery [17]	m^2	m^2-1	$T_A+\lceil\log_2(2m-k-1)\rceil T_X$
KA [14]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4}+4m+k-\frac{23}{4}$ (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[18]	$\frac{m^2}{2}+(m-k)^2$	$\frac{m^2}{2}+(m-k)^2+2k$	$T_A+(2+\lceil\log_2(m-1)\rceil)T_X$
Modified KA[13]	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[20]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4}+O(m\log_2 m)$ (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[31]	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 [21]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4}+\frac{m}{2}+O(m\log_2 m)$ (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 [33] $m=nk$	$\frac{m^2}{2}+\frac{mk}{2}$	$\frac{m^2}{2}+\frac{mk}{2}+\frac{5mn}{4}+O(m\log_2 k)$	$T_A+(\lceil\log_2 k\rceil+\lceil\log_2 3n\rceil)T_X$
This paper (optimal)	$\frac{m^2}{2}+O\left(\frac{\sqrt{11}m^{3/2}}{4}\right)$	$\frac{m^2}{2}+O\left(\frac{\sqrt{11}m^{3/2}}{2}\right)$	$\leq T_A+(2+\lceil\log_2(\ell+3)\rceil$ $+\lceil\log_2(n+2)\rceil)T_X$
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 and proposed a new type of $GF(2^m)$ multiplier architecture. By investigating the choice of the KA parameters, we give the explicit space and time complexity formulations. As a main contribution, the space complexity of our proposal can achieve to $O(\frac{m^2}{2}+\frac{m^{3/2}}{4})$, which is lower than current hybrid multipliers. Meanwhile, its time complexity is less than $T_A+(2+\lceil\log_2(\ell+3)\rceil+\lceil\log_2(n+2)\rceil)T_X$. To find more special type of trinomial that can lead to a better space and time complexity trade-off is the future work.

APPENDIX A
PROOFS

A. Proof of Lemma 1

Proof The proof of this lemma mainly utilizes the reduction formulation (11). 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 $-k \le \Delta < 0$, then $0 < -\Delta < k$, 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 3.3.1

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

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

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

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 $\ell \ge k$ and $\Delta_i = (n-2i+1)\ell - k$ (n even), $\Delta_i = (n-2i)\ell - k$ (n odd), we have all the Δ_is are greater than 0. Therefore, the modular reduction of $p_2^{(i)} x^{\Delta_i - k}$ is given by:

$$\begin{aligned} p_2^{(i)} x^{\Delta_i-k} \bmod f(x) = &\sum_{j=0}^{m-1} h_j^{(i)} x^{-k+(j+\Delta_i) \bmod m} \\ &+ \sum_{j=m-\Delta_i}^{m-1} h_j^{(i)} x^{j+\Delta_i-m}, \end{aligned} \tag{21}$$

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 $i \ne j$. Two independent expressions $\sum_{i=1}^{\lambda} p_1^{(i)} x^{\Delta_i}$ and $\sum_{i=1}^{\lambda} p_1^{(i)} x^{\Delta_i - k}$ can be implemented in parallel. Plus n expressions in (21), we immediately conclude the proposition.

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

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

But if $n \ge 5$, we have $(n-2)\ell \ge \frac{(n+1)\ell}{2} > k$. Therefore, Case 4 only happens if $n = 3$. 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:

$$p_2^{(1)} x^{\ell-2k} \bmod f(x) = \sum_{j=0}^{m-1} h_j^{(i)} x^{-k+(j+\ell-k) \bmod m} + \sum_{j=0}^{k-\ell-1} h_j^{(i)} x^{j+\ell-k}, \tag{22}$$

and

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

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

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

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

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

Similar with Case 2, one can easily check that the subexpressions of (25) have no overlapped terms with each other. However, it is possible here $p_{1,1}^{(\frac{n-t}{2})}$ is overlapped with $p_{1,1}^{(\frac{n-t}{2})} x^{-k}$, which can not be concatenated together. But $\sum_{i=1}^{\frac{n-1}{2}} p_1^{(2)} x^{\Delta_i - k} \bmod f(x)$ consist of at most $2 \cdot \frac{n-1}{2} = n-1$ subexpressions. Meanwhile, some of these subexpressions have no overlapped term with each other. Thus, it totally requires $\lceil \log_2(\frac{n-1}{2} + 3 + \max\{\frac{n-t}{2} - 1, \frac{t+1}{2}\}) \rceil T_X \leq \lceil \log(n+2) \rceil$ delay in parallel. $\square$

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," in IET Information Security, vol. 6, no. 1, pp. 14-19, March 2012.
[8] Y. Chen. "On Space-Time Trade-Off for Montgomery Multipliers over Finite Fields," MD thesis, Department of Computer Science and operational Research, Montreal University, Montreal, Canada. 2015. https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/12571/Chen_Yiyang_2015_memoire.pdf
[9] K.K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley & Sons, 1999.
[10] F. Rodríguez-Henríquez and Ç.K. Koç, "On fully parallel Karatsuba multipler for $GF(2^m)$," in *Proc. Int. Conf. Computer Science and Technology (CST 2003)*, ATA Press, 2003, pp. 405–410.
[11] J. Von Zur Gathen and J. Shokrollahi, "Efficient FPGA-based Karatsuba multipliers for polynomial over $\mathbb{F}_2$," in *Proc 12th Workshop on Selected Areas in Cryptography (SAC 2005)*, Springer 2006, pp. 359–359.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] 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.
[22] Christophe Negre. "Efficient parallel multiplier in shifted polynomial basis," *J. Syst. Archit.*, vol. 53, no. 2-3, pp. 109–116, 2007.
[23] Francisco Rodríguez-Henríquez and Çetin Kaya Koç. "Parallel multipliers based on special irreducible pentanomials," *IEEE Trans. Comput.*, vol. 52, no. 12, pp. 1535–1542, 2003.
[24] 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.
[25] B. Sunar and Ç.K. Koç, "Mastrovito multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 48,no. 5, pp. 522–527, 1999.
[26] A. Halbutogullari and Ç.K. Koç, "Mastrovito multiplier for general irreducible polynomials," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 503–518, May 2000.
[27] 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.
[28] 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.
[29] H. Wu. "Bit-parallel finite field multiplier and squarer using polynomial basis," *IEEE Trans. Comput.*, vol. 51, no. 7, pp. 750–758, 2002.
[30] H. Wu. "Montgomery multiplier and squarer for a class of finite fields," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 521–529, 2002.
[31] 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.
[32] 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.
[33] 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.
[34] 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.