Efficient Bit-Parallel Multiplier for All Trinomials Based on $n$-Term Karatsuba Algorithm

SUN-MI PARK, KU-YOUNG CHANG, DOWON HONG, AND CHANGHO SEO

1Department of Applied Mathematics, Kongju National University, Gongju 32588, South Korea
2Cyber Security Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea

Corresponding author: Changho Seo (chseo@kongju.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant 2019R1A2C1003146, and in part by the Electronics and Telecommunications Research Institute (ETRI) Grant funded by the Korean Government (Core Technology Research on Trust Data Connectome) under Grant 20ZR1300.

ABSTRACT Recently, hybrid multiplication schemes over the binary extension field $GF(2^m)$ based on $n$-term Karatsuba algorithm (KA) have been proposed for irreducible trinomials. Their complexities depend on a decomposition of $m$ and the choice of a generation polynomial. However, these multipliers have some limitations on a decomposition of $m$ or generation polynomial $x^m + x^k + 1$ such that $m \geq 2k$. In this paper, we loosen such limited conditions. We present a new hybrid bit-parallel multiplier based on $n$-term KA for any irreducible trinomial $x^m + x^k + 1$ ($0 < k < m$), where $m$ is decomposed as $m = nm_0 + r$ with $0 < r < m_0$ and $1 < n$. (Here, various values for $n, m_0$ and $r$ may be chosen.) To this end, we generalize the previously proposed multiplication scheme for $x^{nm_0+1} + x^k + 1$ into $x^{nm_0+r} + x^k + 1$. We evaluate the explicit complexity of the proposed multiplier. Specific comparisons show that the proposed multiplier achieves the lowest space complexity with the same or lower time complexity among hybrid multipliers. Compared to the fastest multipliers, the time complexity of the proposed multiplier costs only $T_X$ higher while its space complexity is much lower (it has roughly 40% reduced space complexity), where $T_X$ is the delay of one 2-input XOR gate.

INDEX TERMS Hybrid bit-parallel multiplier, trinomial, Karatsuba algorithm, Mastrovito approach, shifted polynomial basis.

I. INTRODUCTION Efficient hardware implementations of the binary extension field $GF(2^m)$ arithmetic are desired for various areas such as elliptic curve cryptography, computer algebra, and error correcting code ([1]–[3]). For example, the main operation in elliptic curve cryptography is a point multiplication $kP$, where $k$ is an integer and $P$ is a point on elliptic curves. To efficiently implement such point multiplication on hardware, efficient finite field arithmetic are required. Particularly, field multiplication is considered as the most important building block. For this reason, a number of multiplication algorithms have been presented. The space and time complexities are two major factors to measure the efficiency of a multiplication algorithm. The space complexity is represented as the total number of XOR gates and AND gates used. The corresponding time complexity is defined as the total delay of the circuit implementing the multiplier, which is expressed in terms of $T_X$ (the delay of an XOR gate) and $T_A$ (the delay of an AND gate). Such complexities mainly depend on the choice of field basis and a generation polynomial. Multiplication designs over $GF(2^m)$ defined by an irreducible trinomial using the polynomial basis (PB) usually have lower complexities than other choices.

Multiplication architectures over $GF(2^m)$ using PB is often accomplished in two steps, which are polynomial multiplication and modular reduction. There has been a lot of efforts to efficiently perform these two steps. For instance, divide-and-conquer approaches such as Karatsuba algorithm (KA), Winograd short convolution algorithm, and Chinese remainder theorem are used to reduce the space complexity of the polynomial multiplication step ([4]–[6]). However, such methods generally cause an increase in time complexity. For an efficient modular reduction step, a shifted polynomial basis (SPB), which is a variation of the polynomial basis, is adopted in [7], [8]. Mastrovito in [9] proposed Mastrovito
In this paper, we would like to further ease limitation on trinomials and a decomposition of $m$ in [18], [19]. We present a hybrid multiplier for any irreducible trinomial $x^m + x^k + 1$ ($0 < k < m$) with a decomposition $m = nm_0 + r$ with $0 < r < m_0$ and $1 < n$. (Since the case $r = 0$ is already dealt with in [18, Section III], we do not address the case.) To this end, we generalize the multiplication scheme for trinomial $x^{nm_0+1} + x^k + 1$ in [18] into $x^{nm_0+r} + x^k + 1$, which combines $n$-term KA and Mastrovito approach. We evaluate the explicit space and time complexities of the proposed multiplier. The complexity of the proposed multiplier is comparable with those in [18] and [19]. However, more flexible options for trinomials and values for $n, m_0$ and $r$ may yield better complexities. We show this by giving some specific comparisons for several fields containing ones recommended by NIST. The proposed multiplier achieves the lowest space complexity with the same or lower time complexity among hybrid multipliers. Compared to the fastest multipliers, the time complexity of the proposed multiplier costs $T_X$ higher while its space complexity is much lower (it has roughly 40% reduced space complexity).

The rest of this paper is organized as follows. In Section II, we introduce some notations and review the multiplication scheme for trinomial $x^{nm_0+1} + x^k + 1$ presented in [18]. In Section III, we generalize the multiplication scheme for $x^{nm_0+1} + x^k + 1$ into any trinomial $x^{nm_0+r} + x^k + 1$. We evaluate the explicit complexity of the presented multiplier. Section IV gives a comparison of the proposed multiplier with the best-known similar multipliers for trinomials. Finally, some conclusions are drawn.

II. MULTIPLICATION SCHEME FOR TRINOMIAL $x^{nm_0+1} + x^k + 1$ BASED ON $n$-TERM KA

Let the binary extension field $GF(2^m)$ be generated by an irreducible trinomial $x^m + x^k + 1$ with $0 < k < m$. In order to represent field elements of $GF(2^m)$, a shifted polynomial basis is used, which is defined as follows.

**Definition 1** [7]: Let $v$ be an integer and the ordered set $\{1, x, \cdots, x^{m-1}\}$ be a polynomial basis of $GF(2^m)$ over $GF(2)$. The ordered set $x^{-v}\{x^{-v}, x^{1-v}, \cdots, x^{m-1-v}\}$ is called the shifted polynomial basis (SPB) with respect to $\{1, x, \cdots, x^{m-1}\}$.

Here, the SPB $\{x^{-k}, x^{1-k}, \cdots, x^{m-1-k}\}$ is chosen to represent field elements of $GF(2^m)$ for an efficient modular reduction ([7]).

First, we introduce some notations which are used throughout this paper. For a matrix $A$,

- $A_i$ := the $i$th row vector of $A$
- $A[\uparrow i]$ := up and down shifts of $A$ by $i$ rows and feeding the vacancies with zero
- $A[\downarrow i]$ := the lower $i$ rows of $A$
- $A[\circ i]$ := the matrix whose rows are cyclic shift of $A$ by upper $i$ rows

In this paper, we consider a hybrid bit-parallel multiplier for $GF(2^m)$ defined by an irreducible trinomial $x^m + x^k + 1$ ($0 < k < m$). The fastest bit-parallel multiplication architectures for trinomials were proposed in [8], [11]. These multipliers are quadratic multipliers, which have the complexity: $m^2$ AND and $m^2 - 1$ XOR gates, and delay $T_A + (1 + \lceil \log_2 m \rceil)T_X$ (for good field, it is equal to $T_A + \lceil \log_2 m \rceil T_X$). There are many hybrid multipliers for trinomials ([6], [13]–[16]). They have at most about 25% reduced space complexities compared to the fastest bit-parallel multipliers, but cost several more XOR gates delay.

In recent years, Li et al. in [17] proposed a hybrid Karatsuba-based multiplier for a special class of a trinomial $x^m + x^k + 1$ with $m = nk$. The multiplier applies $n$-term KA once in the polynomial multiplication to reduce the space complexity. Also, it uses SPB and Mastrovito approach for the efficient time complexity. Its lowest asymptotic space complexity is $O\left(\frac{1}{2}m^2 + m^2\right)$ circuit gates while its time complexity is only $T_X$ slower than the fastest multipliers. However, irreducible trinomials $x^m + x^k + 1$ are not abundant. Park et al. [18] extended the multiplication scheme in [17] into trinomials $x^{nm_0} + x^k + 1$ such that $m$ is decomposed as $m = nm_0$ or $m = nm_0 + 1$ with $n, m_0 > 1$. The complexity of this multiplier depends on the choices of values $n, m_0$, and $k$. Although an arbitrary integer $m$ can be decomposed into $m = nm_0$ or $m = nm_0 + 1$ with $n, m_0 > 1$, the options for values of $n, m_0$ are limited. Such limitation for the decomposition of $m$ is loosened in [19]. Li et al. in [19] utilize alternative approach to perform modular reduction instead of Mastrovito approach and presented $n$-term Karatsuba multiplier for a trinomial $x^{nm_0+r}+x^k+1$, where $m = nm_0 + r$ with $0 \leq r < n, m_0$. However, there is the limitation $m \geq 2k$ for the choice of trinomial. The space complexity of the multiplier roughly matches that of [18], while its time complexity is slightly higher. However, flexible parameters $n, m_0, r$ may yield lower space complexities or time complexities compared to [18].
\( \mathbf{A}^T \) := the transpose of \( \mathbf{A} \)
- \( \mathbf{O}_{ij} := \) the \( i \times j \) zero matrix

For example, we have

\[
\begin{bmatrix}
\mathbf{A}_{3,*} \\
\mathbf{A}_{4,*} \\
\vdots \\
\mathbf{A}_{m,*} \\
\mathbf{A}_{1,*} \\
\mathbf{A}_{2,*} \\
\mathbf{A}_{3,*}
\end{bmatrix}, \quad \begin{bmatrix}
\mathbf{A}_{3,*} \\
\mathbf{A}_{4,*} \\
\vdots \\
\mathbf{A}_{m,*} \\
\mathbf{A}_{1,*} \\
\mathbf{A}_{2,*} \\
\mathbf{A}_{3,*}
\end{bmatrix}, \quad \begin{bmatrix}
\mathbf{A}_{1,*} \\
\mathbf{A}_{2,*} \\
\mathbf{A}_{3,*}
\end{bmatrix}
\]

Now, we briefly describe the multiplication scheme for trinomial-based on \( n \)-term KA presented in [18]. Let the degree \( m \) of \( GF(2^m) \) be decomposed as \( m = nm_0 + 1 \) with \( n, m_0 > 1 \).

The values of \( n \) and \( m_0 \) can be flexibly chosen. Two given elements \( A = x^{-k} \sum_{i=0}^{n-1} a_i x^i \) and \( B = x^{-k} \sum_{i=0}^{n-1} b_i x^i \) of \( GF(2^m) \) in the SPB \( \{x^{-k}, x^{-k+1}, \ldots, x^{-m+1} \} \) are split as follows.

\[
A = x^{-k}(A_0 + A_1 x^{m_0} + \cdots + A_{n-1} x^{(n-1)m_0} + a_{nm_0} x^{nm_0})
\]

\[
B = x^{-k}(B_0 + B_1 x^{m_0} + \cdots + B_{n-1} x^{(n-1)m_0} + b_{nm_0} x^{nm_0})
\]

where \( A_i = \sum_{j=0}^{m_0-1} a_{ij} x^{j} \) and \( B_i = \sum_{j=0}^{m_0-1} b_{ij} x^{j} \) for \( 0 \leq i < n \). Then, the polynomial multiplication \( AB \) is equal to

\[
AB = \left( \sum_{i=0}^{n-1} A_i x^{im_0} \right) \left( \sum_{i=0}^{n-1} B_i x^{im_0} \right)
\]

\[
+ a_{nm_0} x^{nm_0} \left( \sum_{i=0}^{n-1} B_i x^{im_0} \right) + b_{nm_0} x^{nm_0} \left( \sum_{i=0}^{n-1} A_i x^{im_0} \right)
\]

\[
+ a_{nm_0} b_{nm_0} x^{2nm_0} x^{-2k}
\]

The above product is expanded by using the following \( n \)-term KA.

**Lemma 1** ([12] and [17, Section II-C] or [18, Lemma 2]):

Let \( \sum_{i=0}^{n-1} a_i x^i \) and \( \sum_{i=0}^{n-1} b_i x^i \) be two polynomials in \( GF(2^m)[x] \). Then the multiplication of the two polynomials is equal to

\[
(\sum_{i=0}^{n-1} a_i x^i)(\sum_{i=0}^{n-1} b_i x^i)
\]

\[
= \left( e_{n-1} x^{n-1} + e_{n-2} x^{n-2} + \cdots + e_0 \right)
\]

\[
\times \left( x^{n-1} + x^{n-2} + \cdots + 1 \right)
\]

\[
+ \sum_{i=0}^{2n-3} \sum_{u+v=i, 0 \leq u < v < n} e_{u,v} x^i,
\]

where \( e_j := a_j b_j \) for \( 0 \leq j < n \) and \( e_{u,v} := (a_u + a_v)(b_u + b_v) \) for \( 0 \leq u < v < n \). By using the above lemma, the multiplication \( \left( \sum_{i=0}^{n-1} A_i x^{im_0} \right) \left( \sum_{i=0}^{n-1} B_i x^{im_0} \right) \) is arranged as

\[
\left( \sum_{i=0}^{n-1} A_i x^{im_0} \right) \left( \sum_{i=0}^{n-1} B_i x^{im_0} \right)
\]

\[
= \left( E_{n-1} x^{(n-1)m_0} + E_{n-2} x^{(n-2)m_0} + \cdots + E_0 \right) K(x)
\]

\[
+ \sum_{i=0}^{2n-3} \sum_{u+v=i, 0 \leq u < v < n} E_{u,v} x^{im_0}
\]

and the product \( AB \) is partitioned into three parts

\[
AB = S_1 + S_2 + S_3,
\]

where

\[
E_i = A_i B_i \text{ for } 0 \leq i < n,
\]

\[
E_{u,v} = (A_u + A_v)(B_u + B_v) \text{ for } 0 \leq u < v < n,
\]

\[
K(x) = x^{(n-1)m_0} + x^{(n-2)m_0} + \cdots + 1,
\]

\[
S_1 = \left( \sum_{i=0}^{n-1} E_i K(x) x^{im_0} \right) x^{-2k},
\]

\[
S_2 = \left( \sum_{i=1}^{2n-3} \sum_{u+v=i, 0 \leq u < v < n} E_{u,v} x^{im_0} \right) x^{-2k},
\]

\[
S_3 = \left( \sum_{i=0}^{n-1} (a_{nm_0} B_i + b_{nm_0} A_i) x^{im_0} \right.
\]

\[
+ a_{nm_0} b_{nm_0} x^{2nm_0} x^{-2k}).
\]

Therefore, the field multiplication \( C = AB \) mod \( x^{nm_0 + 1} + x^k + 1 \) is equal to

\[
C = (S_1 + S_2 + S_3) \text{ mod } x^{nm_0 + 1} + x^k + 1.
\]

**A. Computation of \( S_1 \) mod \( x^{nm_0 + 1} + x^k + 1\)**

The computation of \( S_1 \) mod \( x^{nm_0 + 1} + x^k + 1 \) in (1) is performed based on Mastrovito approach ([9]). That is, \( S_1 \) mod \( x^{nm_0 + 1} + x^k + 1 \) is represented as a matrix-vector product

\[
S_1 \text{ mod } x^{nm_0 + 1} + x^k + 1 = \mathbf{Mb}
\]

for a matrix \( \mathbf{M} \) and vector \( \mathbf{b} \). (\( \mathbf{M} \) is called Mastrovito matrix corresponding to \( S_1 \) mod \( x^{nm_0 + 1} + x^k + 1 \).) To this end, the polynomial multiplication \( A_i B_i \) is first implemented as the matrix-vector product

\[
\begin{bmatrix}
\mathbf{A}_{i,L} \\
\mathbf{A}_{i,H}
\end{bmatrix} \mathbf{b}.
\]

where \( \mathbf{b} \) is the coefficient vector of \( B_i, \mathbf{A}_{i,L} \) and \( \mathbf{A}_{i,H} \) are \( m_0 \times m_0 \) triangular matrices given by

\[
\mathbf{A}_{i,L} = \begin{bmatrix}
a_{im_0+0} & 0 & \cdots & 0 \\
a_{im_0+1} & a_{im_0+0} & 0 & \cdots \\
\vdots & \vdots & \ddots & \vdots \\
a_{im_0+m_0-1} & a_{im_0+m_0-2} & \cdots & a_{im_0+0}
\end{bmatrix},
\]

\[
\mathbf{A}_{i,H} = \begin{bmatrix}
0 & 0 & \cdots & a_{im_0+1} \\
0 & 0 & \cdots & a_{im_0+2} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & a_{im_0+m_0-1}
0 & 0 & \cdots & 0
\end{bmatrix}.
\]

Then, the polynomial \( E_i K(x) = A_i B_i K(x) = (A_j + A_j x^{m_0} + \cdots + A_j x^{(n-1)m_0}) B_i \) is written as the matrix-vector product

\[
\begin{bmatrix}
\mathbf{A}_{i,L} \\
\mathbf{A}_{i,L} + \mathbf{A}_{i,H}
\end{bmatrix} \mathbf{b}.
\]

Therefore, the polynomial

\[
S_1 = x^{-2k} \left( \sum_{i=0}^{n-1} E_i K(x) x^{im_0} \right)
\]
can be expressed as big matrix-vector product $\mathbf{Ab}$, where $\mathbf{A}$ is a $2nm_0 \times nm_0$ matrix and $\mathbf{b}$ is an $nm_0 \times 1$ vector defined in (3), as shown at the bottom of the next page. The $i$th row $\mathbf{A}_{i,*}$ of $\mathbf{A}$ corresponds to the coefficient of $x^{i−1−2k}$ of $S_1$. Finally, the Mastrovito matrix $\mathbf{M}$ related to $\mathbf{Ab}$ mod $x^{nm_0+1}+x^k+1$ is obtained by reducing terms of degrees that are out of the range $[-k, m - k - 1]$ by $x^{nm_0+1}+x^k+1$.

**B. COMPUTATION OF $S_2 \mod x^{nm_0+1}+x^k+1$**

The polynomial $S_2$ in (1) is split into some parts using the following lemma.

**Lemma 2 ([18, Lemma 4] or [20, Proposition 1]):** The polynomial $S_2$ in (1) can be partitioned as follows:

$$
\sum_{i=1}^{2n-3} \left( \sum_{u+v=i, 0 \leq u < v < n} E_{u,v} x^{im_0} \right) = \sum_{i=1}^{\lambda} G_i x^{(2^i-2i+1)m_0}
$$

where $\lambda = \left\lceil \frac{n}{2} \right\rceil$, $G_i = \sum_{j=0}^{2^i-2i} C_j^{(i)} x^{jm_0}$, and $C_j^{(i)} \in \{E_{u,v}\} | 0 \leq u < v < n$.

From the above lemma, we can write $S_2$ as

$$
S_2 = \sum_{i=1}^{\lambda} G_i',
$$

where $G_i' = G_i x^{(2^i-2i+1)m_0-2k}$ for $1 \leq i \leq \lambda$. Also, the polynomial $G_i = G_i x^{(2^i-2i+1)m_0-2k}$ can be written as

$$
G_i' = \sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k} u_j^{(i)} x^j \quad \text{if } n \text{ is even,}
$$

$$
G_i' = \sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k} u_j^{(i)} x^j \quad \text{if } n \text{ is odd}
$$

with $u_j^{(i)} \in GF(2)$ since $\lambda = \lceil n/2 \rceil$ and the degrees of terms of $G_i$ are in the range

$$
\begin{align*}
\{0, nm_0 - 2\} & \quad \text{if } n \text{ is even,} \\
\{0, nm_0 + m_0 - 2\} & \quad \text{if } n \text{ is odd}
\end{align*}
$$

by the definition of $G_i$ in Lemma 2. According to [18, Section III-B], the computation of polynomials $G_i$’s (i.e., $G_i$’s) is performed with the complexity

$$
\begin{align*}
\text{Delay} &= T_A + (1 + \lceil \log_2 m_0 \rceil) T_X, \\
\text{AND} &= \frac{1}{2} n^2 m_0^2 - 1 - nm_0^2, \\
\text{XOR} &= \frac{1}{2} n^2 m_0^2 - \frac{1}{2} m_0^2 + \frac{1}{2} n^2 m_0 - nm_0 \\
&+ \frac{1}{2} n + \begin{cases} 0 & \text{if } n \text{ : even,} \\
\frac{1}{2} m_0 - \frac{1}{2} & \text{if } n \text{ : odd.}
\end{cases}
\end{align*}
$$

Next, $S_2 \mod x^{nm_0+1}+x^k+1 = \sum_{i=1}^{\lambda} (G_i' \mod x^{nm_0+1}+x^k+1)$ is obtained by implementing the modular reductions for $G_1', \ldots, G_\lambda'$ mod $x^{nm_0+1}+x^k+1$ and adding all these results by a binary XOR tree.

### III. GENERALIZED MULTIPLICATION SCHEME FOR TRINOMIAL $x^{nm_0+r}+x^k+1$

In this section, we present a multiplication scheme for any trinomial based on $n$-term KA, which is a generalization of that for trinomial $x^{nm_0+1}+x^k+1$ in the previous section. Let $GF(2^m)$ be defined by an irreducible trinomial $F(x) = x^m + x^k+1 \ (0 < k < m)$ and the SPB $\{x^{-k}, x^{-1-k}, \ldots, x^{m-1-k}\}$ be used to represent a field element. We decompose $m$ as $m = nm_0 + r$, where $1 \leq r < m_0$ and $1 < n$. Here, a flexible decomposition of $m$ is possible, that is, various values for $n$, $m_0$, and $r$ may be chosen. Throughout this paper, we define

$$
t := \left\lceil \frac{k}{m_0} \right\rceil
$$

Then we have that $(t-1)m_0 < k \leq tm_0$ and $1 \leq t \leq n + 1$.

We first split two given elements $A = x^{−k} \sum_{i=0}^{m_0−1} a_i x^i$ and $B = x^{−k} \sum_{i=0}^{m_0−1} b_i x^i$ of $GF(2^m)$ in the SPB $\{x^{-k}, x^{-1-k}, \ldots, x^{m-1-k}\}$ as follows.

$$
A = x^{−k}(A_0 + A_1 x^{m_0} + \ldots + A_{t−1} x^{(n−1)m_0} + \tilde{A}^{mn_0}),
$$

$$
B = x^{−k}(B_0 + B_1 x^{m_0} + \ldots + B_{t−1} x^{(n−1)m_0} + \tilde{B}^{mn_0}),
$$

where

$$
\tilde{A} = \sum_{j=0}^{r−1} a_{j+m_0} x^j, \quad \tilde{B} = \sum_{j=0}^{r−1} b_{j+m_0} x^j,
$$

$$
A_i = \sum_{j=0}^{m_0−1−j} a_{j+m_0} x^j, \quad B_i = \sum_{j=0}^{m_0−1−j} b_{j+m_0} x^j
$$

for $0 \leq i < n$. Then, the product $AB$ equals to

$$
AB = (\left(\sum_{i=0}^{n−1} A_i x^{im_0}\right)\left(\sum_{i=0}^{n−1} B_i x^{im_0}\right) + \tilde{A}^{mn_0} B_i x^{im_0} + \sum_{i=0}^{n−1} B_i x^{im_0})
$$

$$
+ \tilde{B}^{mn_0} (\sum_{i=0}^{n−1} A_i x^{im_0}) + \tilde{A} \tilde{B}^{2mn_0} x^{−2k}.
$$

We expand the term $(\sum_{i=0}^{n−1} A_i x^{im_0}) (\sum_{i=0}^{n−1} B_i x^{im_0})$ using $n$-term KA in Lemma 1, and partition the product $AB$ into three parts

$$
AB = S_1 + S_2 + S_3,
$$

where

$$
S_1 = (\sum_{i=0}^{n−1} E_i K(x) x^{im_0}) x^{−2k},
$$

$$
S_2 = (\sum_{i=0}^{2n−3} \left(\sum_{u+v=i, 0 \leq u < v < n} E_{u,v} x^{im_0}\right) x^{−2k},
$$

$$
S_3 = (\sum_{i=0}^{n−1} B_i x^{im_0} \tilde{A} + (\sum_{i=0}^{n−1} A_i x^{im_0}) \tilde{B}) + \tilde{A} \tilde{B} x^{2mn_0} x^{−2k},
$$

$$
E_i = A_i B_i \text{ for } 0 \leq i < n,
$$

$$
E_{u,v} = (A_u + A_v) (B_u + B_v) \text{ for } 0 \leq u < v < n,
$$

$$
K(x) = x^{(n−1)m_0} + x^{(n−2)m_0} + \ldots + 1.
$$

Therefore, the field multiplication $C = AB \mod F(x)$ is equal to

$$
C = (S_1 + S_2 + S_3) \mod F(x).
$$

It is noted that the polynomials $S_1$ and $S_2$ have the same forms as those in (1), respectively, except for the reduction
polynomial. Fig. 1 (a) depicts the architecture of the field multiplication $C = AB \mod F(x)$. Here, Addition block implements the additions of vectors in Table 3 for $M_1 \mathbf{b}$ and $M_2 \mathbf{b}$, where $S_1 \mod F(x) = M_1 \mathbf{b} + M_2 \mathbf{b}$. Fig. 1 (b) illuminates detailed structures of AND blocks and BTX blocks in Fig. 1 (a), which is the same as that in [21, Fig. 1 (b)]. Fig. 2 (given in Section III-B2) shows the architecture for $Q_1, \cdots, Q_w$, where $S_3 \mod F(x) = Q_1 + \cdots + Q_w$, (see (15)). We expound the computation of $C = (S_1 + S_2 + S_3) \mod F(x)$.

**A. COMPUTATION OF $S_1 \mod F(x)$**

1) **MASTROVITO MATRIX CORRESPONDING TO $S_1 \mod F(x)$**

The polynomial $S_1 = (\sum_{i=0}^{m-1} E_i K(x)x^{im_0})x^{-2k}$ in (5) can be written as the matrix vector product $Ab$ as in Section II-A, where $A$ is the $(2m - 2r) \times (m - r)$ matrix and $b$ is the vector given in (3). The $i$th row $A_{i, \ast}$ of $A$ corresponds to the coefficient of $x^{i - 1 - 2k}$ of $S_1$ and so the degrees of terms of $S_1$ are in the range $[-2k, 2m - 2k - 2r - 2]$. (The last row $A_{2m-2r, \ast}$ of $A$ is a zero row.)

Now, we compute the Mastrovito matrix $M$ corresponding to $S_1 \mod F(x)$ by reducing terms of $S_1$ of degrees that are out of the range $[-k, m - k - 1]$. We use the following reduction rules

$$x^i = \begin{cases} x^{i+k} + x^{i+m} & \text{for } -2k \leq i < -k, \\ x^{i-m} + x^{i-m+k} & \text{for } m - k \leq i < 2m - 2k. \end{cases} \quad (6)$$

We express modular reductions as additions of some matrices. To this end, we first define the $m \times (m - r)$ matrices $R_1$, $R_2$, and $R_3$ according to the value of $k$ in Table 1 (0 denotes a zero row). The $i$th row of $R_1$ corresponds to the coefficient of $x^{i - 1 - k}$ of $S_1$. The $k$ terms $x^{-2k}, x^{-2k+1}, \ldots, x^{-1}$ (whose coefficients correspond to the first $k$ rows of $A$) are reduced as

$$x^{-2k} = x^{-k} + x^{-m-2k}$$

$$\vdots$$

$$x^{-k-1} = x^{-1} + x^{-m-k-1}$$

mod $F(x)$

TABLE 1. Definitions of $m \times (m - r)$ matrices $R_1$, $R_2$, and $R_3$.

|               | If $k < m - 2r$ | If $k \geq m - 2r$ |
|---------------|-----------------|-------------------|
| $R_1$         | $A_{k+1, \ast}$ | $A_{k+1, \ast}$   |
|               | $\vdots$        | $\vdots$          |
|               | $A_{m+k, \ast}$ | $A_{m+k, \ast}$   |
| $R_2$         | $A_{1, \ast}$   | $A_{1, \ast}$     |
|               | $\vdots$        | $\vdots$          |
|               | $A_{2m-2r, \ast}$ | $A_{2m-2r, \ast}$ |
|               | $m-k$           | $m-k$             |
| $R_3$         | $0$             | $1$               |
|               | $\vdots$        | $\vdots$          |
|               | $0$             | $0$               |
|               | $k$             | $m-k$             |

by (6). The above equations show that terms of $S_1$ of degrees in $[-2k, -k - 1]$ are reduced by adding $R_2$ and $R_2 \otimes k$ to $R_1$. If $k \geq m - 2r$, then there are no more terms to be reduced. Otherwise $k < m - 2r$, the terms of $S_1$ of degrees in $[m - k, 2m - 2k - 2r - 2]$ are reduced as

$$x^{m-k} = x^{-k} + x^0$$

$$\vdots$$

$$x^{2m-2k-2r-2} = x^{m-2k-2r-2} + x^{m-k-2r-2}$$

mod $F(x)$

$$R_3 \otimes k$$

$$R_1$$

by (6) and such reductions are implemented by adding $R_3$ and $R_3 \otimes k$ to $R_1$. (We note that $A_{2m-2r, \ast}$ is a zero row.) Hence, the Mastrovito matrix $M$ corresponding to $S_1 \mod F(x)$ is

$$A := \left[ \begin{array}{cccc} A_{0, L} & O_{m_0 \times m_0} & \cdots & O_{m_0 \times m_0} \\ A_{0, L} + A_{0, H} & A_{1, L} & \cdots & O_{m_0 \times m_0} \\ \vdots & A_{1, L} + A_{1, H} & \cdots & O_{m_0 \times m_0} \\ A_{0, H} & A_{1, L} + A_{1, H} & \cdots & O_{m_0 \times m_0} \\ O_{m_0 \times m_0} & A_{1, H} & \cdots & A_{n-2, H} \\ \vdots & \vdots & \vdots & \vdots \\ O_{m_0 \times m_0} & O_{m_0 \times m_0} & \cdots & O_{m_0 \times m_0} \end{array} \right]$$

$$n-1$$

$$\mathbf{b} := [b_0^T, b_1^T, \cdots, b_{n-1}^T]^T \quad (3)$$

$$n-1$$
equal to

\[ M = R_1 + (R_2 + R_3) \oplus k + (R_2 + R_3) = M_1 + M_2, \]

where

\[
M_1 = R_1 + (R_2 + R_3) \oplus k = (M_1' + M_1'') \oplus k, \\
M_2 = (R_2 + R_3).
\]

Consequently, we have that

\[ S_1 \mod F(x) = M_1 b + M_2 b. \]

2) DESCRIPTION OF TWO VECTORS \(M_1 b\) AND \(M_2 b\)

We give full descriptions of two vectors \(M_1 b\) and \(M_2 b\). From definition of \(A\) in (3), two \(m \times (m-r)\) matrices \(M_1'\) and \(M_1''\) are given as in (7), as shown at the bottom of the next page since \(m = nm_0 + r\). Here, for the matrix \(M_1''\), we use the following facts:

\[
(A_{L,L} + A_{L,H})[\uparrow r] + (A_{L,L} + A_{L,H})[\downarrow m_0 - r] = (A_{L,L} + A_{L,H})[\oplus r], \\
(A_{L,L} + A_{L,H})[\uparrow r] + A_{L,H}[\downarrow m_0 - r] = A_{L,L}[\uparrow r] + A_{L,H}[\oplus r].
\]

We define the following vectors.

\[
e_0 := A_{0,L} b_0, \quad e_i := A_{i-1,H} b_{i-1} + A_{i,L} b_i \quad \text{for } 1 \leq i < n, \\
e_n := A_{n-1,H} b_{n-1}.
\]
Then we have that
\[(M'_1 + M''_1)b\]
\[
= \begin{bmatrix}
  e_0 + e_1[\uparrow r] + e_2[\bigodot r] + \cdots + e_n[\bigodot r] \\
  e_0 + e_1[\uparrow r] + e_2[\bigodot r] + \cdots + e_n[\bigodot r] \\
  \vdots \\
  e_0 + e_1 + e_2 + \cdots + e_{n-2} + e_{n-1}[\uparrow r] + e_n[\bigodot r] \\
  e_0 + e_1 + e_2 + \cdots + e_{n-2} + e_{n-1} + e_n[\uparrow r] \\
  (e_1 + e_2 + \cdots + e_{n-1} + e_n)[\uparrow r]
\end{bmatrix}
\]
and
\[M_1b = (M'_1 + M''_1)b[\bigodot k].\]

The vector \(M_2b = (R_2 + R_3)b\) is equal to
\[
\begin{bmatrix}
  \text{the first } k \text{ rows of } Ab \\
  0 \\
  \vdots \\
  0 \\
\end{bmatrix}
\]

or
\[
\begin{bmatrix}
  \text{the first } k \text{ rows of } Ab \\
  \vdots \\
  0
\end{bmatrix}
\]

according as \(k < m - 2r\) or \(k \geq m - 2r\). Therefore, \(M_2b\) is given as in Table 2 according to the value of \(k\), where
\[
\varepsilon := \begin{cases} 
(e_{r+1} + \cdots + e_n)[(t+1)m_0 - k - r] & \text{if } k \geq m_0 - r, \\
(e_r + e_{r+1} + \cdots + e_n)[(m_0 - k - r)] & \text{if } k < m_0 - r.
\end{cases}
\]

Here, we use that \((t + 1)m_0 < k \leq m_0, 1 \leq t \leq n + 1,\) and
\[
Ab = \begin{bmatrix}
  e_0 \\
  e_0 + e_1 \\
  \vdots \\
  e_0 + e_1 + \cdots + e_{n-1} \\
  e_1 + e_2 + \cdots + e_n
\end{bmatrix}.
\]

3) COMPLEXITY FOR \(S_1 \mod F(x)\)

From now on, we consider the complexity for \(S_1 \mod F(x) = M_1b + M_2b\). The vectors \(e_0, \cdots, e_n\) in (8) are computed by two blocks: AND blocks and BTX blocks in Fig. 1 (b). Such computations require the complexity
\[
\begin{align*}
\text{Delay} &= T_A + \lceil \log_2 m_0 \rceil T_X, \\
\# \text{ AND} &= nm_0^2, \\
\# \text{ XOR} &= (m_0 - 1)(nm_0 - 1)
\end{align*}
\]

according to [18, Section III-A]. After computing vectors \(e_i\)'s, we perform the computation of two vectors \(M_1b \) and \(M_2b\) using sub-expression sharing technique in Lemma 3.

Lemma 3 ([16], [17]): When two expressions \(a_1 + \cdots + a_l + a_{l+1} + \cdots + a_q\) and \(a_1 + \cdots + a_l + a'_{l+1} + \cdots + a'_q\) with \(l\) common terms are simultaneously computed by binary XOR trees, \(l - W(l)\) XOR gates can be saved by reusing common values, where \(W(l)\) is the Hamming weight of \(l\).

In Table 3, we summarize the required number of XOR gates for the computations of two vectors \(M_1b\) and \(M_2b\) by using the above lemma. For instance, if the vector \(e_0 + \cdots + e_{n-1} + e_n[\uparrow r]\) in (v) is computed with \(m_0n\) XOR gates, we can save \(m_0(n - 1 - W(t - 1))\) XOR gates for the computation of \(e_0 + e_1 + \cdots + e_{r-2}\). Therefore, \(e_0 + e_1 + \cdots + e_{r-2}\)
TABLE 2. The vector $M_2b$ according to the value of $k$. 

| $k$ | $M_2b$ |
|-----|--------|
| $k < m - 2r$ | $\begin{bmatrix} e_0 & \ldots & e_{m-k} \end{bmatrix}$ |
| $m - 2r \leq k \leq m - r$ | $\begin{bmatrix} e_0 & \ldots & e_{m-k} \end{bmatrix}$ |
| $k > m - r$ | $\begin{bmatrix} e_0 & \ldots & e_{m-k} \end{bmatrix}$ |

in (vii) is computed with only $m_0(t - 2) - m_0(t - 1 - W(t - 1)) = m_0(W(t - 1) - 1)$ XOR gates. For the vector $M_1b$, (i), (ii), (iii), (v), and (vi) in Table 3 are computed. For the vector $M_2b$, we need to compute (iv), (vii), and (viii). In the case $k > m - r$, the vector $(e_1 + \cdots + e_m)_{\lfloor k-m_0 \rfloor}$ in $M_2b$ is obtained from the vector $(e_1 + \cdots + e_m)_{\lfloor k-r \rfloor}$ in (ii) since $k-m_0 < r$. The last row in Table 3 gives the total number of XOR gates for the computation of two vectors $M_1b$ and $M_2b$. Their critical path delay is $\lceil \log_2(n+1) \rceil T_X$.

Finally, the sum of the two vectors

$$S_1 \mod x^{m_0+1} + x^k + 1 = M_1b + M_2b$$

is implemented with $\max\{k, m-2r\}$ XOR gates and $T_X$ delay. Consequently, the total complexity for $S_1 \mod F(x)$ is

$$\text{Delay} = T_A + (1 + \lceil \log_2 m_0 \rceil + \lceil \log_2(n+1) \rceil) T_X. \# \text{AND} = nm_0^2, \# \text{XOR} = \left\lfloor \frac{n^2 m_0 + nm_0^2}{4} \right\rfloor + \frac{1}{2} nm_0 + \frac{1}{2} nr + J + r W(n - 1) + \max\{k, m - 2r\}$$

using (9) and Table 3, where $J$ is defined in Table 3.

Example 1: We give a small example to describe the proposed multiplication scheme. We consider the finite field $GF(2^{18})$ generated by the irreducible trinomial $F(x) = x^{18} + x^7 + 1$ with parameters $n = 3, m_0 = 5, r = 3, k = 7$. Then, we have that $r = \lceil \frac{k}{m_0} \rceil = 2$. Field elements of $GF(2^{18})$ are represented using the SPB $(x^{-7}, x^{-6}, \ldots, x^{10})$. Let two elements $A = x^{-7}(\sum_{i=0}^{17} a_i x^i)$ and $B = x^{-7}(\sum_{i=0}^{17} b_i x^i)$ in $GF(2^{18})$ be given. We split two elements $A$ and $B$ as

$$A = x^{-7}(A_0 + A_1 x^5 + A_2 x^{10} + \tilde{A} x^{15}),$$
$$B = x^{-7}(B_0 + B_1 x^5 + B_2 x^{10} + \tilde{B} x^{15}),$$

where

$$\tilde{A} = \sum_{j=0}^{2} a_{15+j} x^j, \quad \tilde{B} = \sum_{j=0}^{2} b_{15+j} x^j,$$

$$A_i = \sum_{j=0}^{4} a_{5i+j} x^j, \quad B_i = \sum_{j=0}^{4} b_{5i+j} x^j$$

for $0 \leq i \leq 2$. According to (5), the field multiplication $C$ of the two elements $A$ and $B$ is equal to

$$C = AB \mod F(x) = S_1 + S_2 + S_3 \mod F(x),$$

where

$$S_1 = (E_0 + E_1 x^5 + E_2 x^{10})(x^{10} + x^5 + 1)x^{-14},$$
$$S_2 = (E_0 x^5 + E_0 x^{10} + E_1 x^{15})x^{-14},$$
$$S_3 = (\sum_{i=0}^{2} B_i x^{5i} \tilde{A}) + (\sum_{i=0}^{2} A_i x^{5i} \tilde{B} + \tilde{A} \tilde{B} x^{15})x,$$
$$E_i = A_i B_i \text{ for } 0 \leq i \leq 2,$$
$$E_{u,v} = (A_u + A_v)(B_u + B_v) \text{ for } 0 \leq u < v \leq 2.$$
TABLE 3. Space complexity for $M_1b$ and $M_2b$, where $m = nm_0 + r$.

| Operation                                                                 | # XOR                                                                 |
|----------------------------------------------------------------------------|----------------------------------------------------------------------|
| (i) $e_n[0:r] + \cdots + e_2[0:r] + e_1[1:r] + e_0$                        | $m_0n$                                                                |
| (ii) $(e_n + e_{n-1} + \cdots + e_1)[r]$                                   | $rW(n - 1) \left( = r(n - 1) - r((n - 1) - W(n - l)) \right)$          |
| (iii) $e_n[0:r] + \cdots + e_{l+1}[0:r] + e_l[1:r] + e_{l-1} + \cdots + e_0$ | $m_0n - (m_0 - r)(n - l + 1 - W(n - l + 1) - W(n - l - l))$             |
| for $2 \leq l \leq \lfloor n/2 \rfloor$                                   | $= m_0l - (m_0 - r) + m_0W(n - l + 1) - r(W(n - l + 1) - W(n - l))$     |
| (iv) if $k < m - 2r$, $e_n = e_{n-1} = \cdots = e_{l+1} = e_l = \cdots = e_0$ | $m_0(W(1) - 1)$                                                      |
| $e_n + e_{n-1}$                                                             | $m_0(W(2) - 1)$                                                      |
| $\vdots$                                                                   |                                                                     |
| $e_n + \cdots + e_{l+1} + e_l[0:r]$                                        | $m_0(W(n - t - 1) - 1)$                                               |
| $e_l + \cdots + e_{l+1}[0:r]$                                              | $(t - 1)m_0 - r(W(t - 1))$                                           |
| (v) $e_0 + \cdots + e_{n-1} + e_n[0:r]$                                    | $m_0n - m_0(l - W(l))$                                               |
| (vi) $e_0 + \cdots + e_{l-1} + e_{l}[0:r] + e_{l+1}[0:r] + \cdots + e_n[0:r]$ | $m_0(W(t - 1) - 1)$                                                 |
| for $\lfloor n/2 \rfloor < l \leq n - 1$                                   | $(k - (t - 1)m_0)(W(t) - 1)$                                         |
| (vii) if $k \leq m - r$, $e_0 = e_{n-1} = \cdots = e_{l+1} = e_l = e_{l-1}$ | $m_0(W(1) - 1)$                                                      |
| $(e_0 + \cdots + e_{l-1})[k:((k-r-1)m_0)]$                                 | $m_0(W(n - (t - 1)) - 1)$                                            |
| (viii) if $k > m - r$, $e_0 = e_{n-1} = \cdots = e_{l+1} = e_l = e_{l-1}$ | $m_0(W(n - 1) - 1)$                                                 |
| $(e_0 + \cdots + e_{l+1})[k:((k-r-1)m_0)]$                                | $m_0(W(n) - 1)$                                                     |

Total number of XOR gates for $M_1b$ and $M_2b$:

$$b_i = \left[b_{2i}, b_{2i+1}, b_{2i+2}, b_{2i+3}, b_{2i+4}\right]^T$$

for $0 \leq i \leq 2$. The computation of $S_1 \bmod F(x)$ is implemented as follows (see the left part of Fig. 1 (a)). First, we note that the following vectors $e_0, \cdots, e_3$ are computed by two blocks in Fig. 1 (b).

1. AND blocks: Perform matrix-vector bitwise multiplications $A_{L} \odot b_i$ and $A_{H} \odot b_i$ for $0 \leq i \leq 2$ in parallel.

For example, we compute that

$$A_{L} \odot b_i = \begin{bmatrix}
    a_{ib_{2i}} & 0 & \cdots & 0 \\
    a_{ib_{2i+1}} & a_{ib_{2i+2}} & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{ib_{2i+4}} & a_{ib_{2i+3}} & \cdots & a_{ib_{2i+4}}
\end{bmatrix}$$

2. BTX blocks: Add all the entries of the same row in matrices $A_{L} \odot b_i$, $A_{0,0,H} \odot b_0 + A_{1,1,L} \odot b_1$, $A_{1,1,H} \odot b_1 + A_{2,2,L} \odot b_2$, and $A_{2,2,H} \odot b_2$ using a binary XOR tree.

The above computations need 75 AND gates, 56 XOR gates, and $T_A + 3T_X$ delays. Next, for the vector $M_1b$, the following summations of vectors are computed using sub-expression sharing technique in Lemma 3:

$$(e_3[0:3] + e_2[0:3]) + (e_1[1:3] + e_0) : 15 \text{ XOR gates}$$

$$(e_3 + e_2)[3] + (e_1)[3] : 3 \text{ XOR gates}$$

$$(e_0 + e_1) + (e_2 + e_3[1:3]) : 15 \text{ XOR gates}$$

$$(e_0 + e_1) + (e_2 + e_3[1:3]) : 10 \text{ XOR gates}$$

We note that the vector $(e_0 + e_1)[3]$ in $M_2b$ can be obtained from the calculation process of $(e_0 + e_1) + (e_2 + e_3[1:3])$ without any costs. Therefore, the computation of two vectors $M_1b$ and $M_2b$ requires 43 XOR gates and $2T_X$ delays (see Table 3). Finally, $S_1 \bmod F(x)$ is obtained by adding two vectors $M_1b$ and $M_2b$ with 12 XOR gates and $T_X$ delay.

In summary, the total complexity for $S_1 \bmod F(x)$ is 75 AND gates, 56+$43+12=111$ XOR gates, and $T_A + 6T_X$ delays (see (10)).

The computation of $S_2 + S_3 \bmod F(x)$ is dealt with in Examples 2 and 3.

B. COMPUTATION OF $S_2 + S_3 \bmod F(x)$

In this section, we consider the computation of $S_2 + S_3 \bmod F(x)$, which is implemented in parallel with the computation of $S_1 \bmod F(x)$.

1) COMPUTATION OF $S_2 \bmod F(x)$

Analogous to Section II-B, we can write $S_2$ in (5) as

$$S_2 = \sum_{i=1}^{\lambda} G'_i.$$
TABLE 4. The modular reductions of $G'_i \mod F(x) = x^{nm_0+r} + x^k + 1$ for $1 \leq i \leq \lambda$, where $(t-1)m_0 < k \leq tm_0$.

\[
\begin{array}{|c|c|c|}
\hline
\text{when } n \text{ is even} & G'_i \mod F(x) = \sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k} u_j^{(i)} x^j & \# \text{ XOR} \\
\hline
1 \leq i < \frac{n}{2} - \frac{1}{2} & \\
\quad P_1^{(i)} = \sum_{j=(n-2i+1)m_0-2k}^{(n-2i+1)m_0-2k-2} u_j^{(i)} x^j + \sum_{j=m-k}^{m-1} u_j^{(i)} x^{j-m} & (n-2i+1)m_0-k-r-1 \\
\quad P_2^{(i)} = \sum_{j=m-k}^{m-1} u_j^{(i)} x^{j-m} & + Y_1 \text{ if } i = \frac{n}{2} - \frac{1}{2} \\
\hline
\text{i = } \frac{n}{2} - \frac{1}{2} & \\
\quad P_1^{(i)} = \sum_{j=(n-2i+1)m_0-2k}^{(n-2i+1)m_0-2k-2} u_j^{(i)} x^j & k-(n-2i+1)m_0 \\
\quad P_2^{(i)} = \sum_{j=m-k}^{m-1} u_j^{(i)} x^{j-m} & \\
\hline
\text{when } n \text{ is odd} & \\
\hline
1 \leq i < \frac{n-1}{2} - \frac{1}{2} & \\
\quad P_1^{(i)} = \sum_{j=(n-2i+1)m_0-2k}^{(n-2i+1)m_0-2k-2} u_j^{(i)} x^j + \sum_{j=m-k}^{m-1} u_j^{(i)} x^{j-m} & (n-2i+2)m_0-k-2r-2 \\
\quad P_2^{(i)} = \sum_{j=m-k}^{m-1} u_j^{(i)} x^{j-m} & + Y_2 \text{ if } i = \frac{n-1}{2} - \frac{1}{2} \\
\quad P_3^{(i)} = \sum_{j=m-k}^{m-1} u_j^{(i)} x^{j-m} & 0 \\
\hline
\end{array}
\]

Here, $P_1^{(i)}$, $P_2^{(i)}$, and $P_3^{(i)}$ are obtained without any costs if the polynomials $G'_i$’s are given (see Table 4). Since such modular reductions $G'_i \mod F(x)$ are performed in the same way as [18] and no special technique is used, we only summarize their results in Table 4. As a result, we have:

\[
S_2 \mod F(x) = \sum_{i=1}^{\lambda} (G'_i \mod F(x))
\]

where $\lambda = \left\lfloor \frac{n}{2} \right\rfloor$ and

\[
G'_i = G_i x^{(2n-2i+1)m_0-2k} = \begin{cases} 
\sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k-2} u_j^{(i)} x^j & \text{if } n \text{ is even,} \\
\sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k-2} u_j^{(i)} x^j & \text{if } n \text{ is odd}
\end{cases}
\]

with $u_j^{(i)} \in GF(2)$ for $1 \leq i \leq \lambda$. We first compute the polynomials $G'_i$’s (i.e., $G_i$’s) with the complexity in (4). The architecture for $G'_i$’s is given in the middle part of Fig. 1 (a) (see Example 2 more details). Next, we perform the modular reductions $G'_i \mod F(x)$ for $1 \leq i \leq \lambda$ and then the summation $\sum_{i=1}^{\lambda} (G'_i \mod F(x))$.

$G'_i \mod F(x)$

where $\lambda = \left\lfloor \frac{n}{2} \right\rfloor$ and

\[
G'_i = G_i x^{(2n-2i+1)m_0-2k} = \begin{cases} 
\sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k-2} u_j^{(i)} x^j & \text{if } n \text{ is even,} \\
\sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k-2} u_j^{(i)} x^j & \text{if } n \text{ is odd}
\end{cases}
\]

with $u_j^{(i)} \in GF(2)$ for $1 \leq i \leq \lambda$. We first compute the polynomials $G'_i$’s (i.e., $G_i$’s) with the complexity in (4). The architecture for $G'_i$’s is given in the middle part of Fig. 1 (a) (see Example 2 more details). Next, we perform the modular reductions $G'_i \mod F(x)$ for $1 \leq i \leq \lambda$ and then the summation $\sum_{i=1}^{\lambda} (G'_i \mod F(x))$. The modular reduction formula for $G'_i \mod F(x)$ is different according to the parity of $n$. In fact, we can write

$G'_i \mod F(x)$

\[
G'_i = G_i x^{(2n-2i+1)m_0-2k} = \begin{cases} 
\sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k-2} u_j^{(i)} x^j & \text{if } n \text{ is even,} \\
\sum_{j=(n-2i+1)m_0-2k}^{(2n-2i+1)m_0-2k-2} u_j^{(i)} x^j & \text{if } n \text{ is odd}
\end{cases}
\]
\[
\sum_{i=1}^{n} p_1^{(i)} + \sum_{i=1}^{n} p_2^{(i)}
\]
if \( n \) : even,
\[
\sum_{i=1}^{n-1} p_1^{(i)} + \sum_{i=1}^{n-1} p_2^{(i)} + \sum_{i=1}^{n-1} p_3^{(i)}
\]
+ \( p_2 \) if \( r \) : even, \( k \leq tm_0 - r - 2 \),
\[
0 \quad \text{otherwise}
\]
if \( n \) : odd. 

The last column of Table 4 denotes the number of XOR gates for each modular reduction. The total number of XOR gates for the summations in (11) is reported in the last rows of Table 4 for each parity of \( n \). By adding it to the space complexity for \( S_2 \) mod \( F(x) \) is

\[
\begin{align*}
\# \text{AND} &= \frac{1}{2} n^2 m_0^2 - \frac{1}{2} nm_0^2, \\
\# \text{XOR} &= \frac{1}{2} n^2 m_0^2 - \frac{1}{2} nm_0^2 + \frac{5}{4} n^2 m_0 - 2nm_0 \\
& - \frac{1}{2} nk + (r + 1)\lfloor \frac{r}{2} \rfloor \\
& + 2\lfloor \frac{r}{2} \rfloor k - r - 2\lfloor \frac{r}{2} \rfloor m_0 + Y_1 \\
& + (2\lfloor \frac{r}{2} \rfloor + \frac{1}{2}) k - \frac{1}{2} (r - 1) - \frac{r + 1}{n} \\
& - (2\lfloor \frac{r}{2} \rfloor + 1\lfloor \frac{r}{2} \rfloor m_0 + Y_2 \\
& \text{if } n \text{ : even},
\end{align*}
\]

(12)

where \( Y_1 \) and \( Y_2 \) are given in Table 4.

**Example 2:** Continuing from Example 1, we consider the computations of

\( S_2 \mod F(x) = (E_0,1x^5 + E_0,2x^{10} + E_1,2x^{15})x^{-14} \mod F(x) \)

Since \( \lambda = [\frac{9}{4}] = 1 \), we have that

\( S_2 = G'_1 = G'_1 x^{-9} = \sum_{j=-9}^{9} u_j^{(1)} x^j \),

where \( G'_1 = E_0,1 + E_0,2x^5 + E_1,2x^{10} \). According to Table 4, the modular reduction of \( S_2 = G'_1 = \sum_{j=-9}^{9} u_j^{(1)} x^j \) by \( F(x) \) is equal to

\( G'_1 \mod F(x) = \sum_{j=-9}^{9} u_j^{(1)} x^j \mod F(x) = P_1^{(1)} + P_2^{(1)} + P_3^{(1)} \),

where

\[
P_1^{(1)} = \sum_{j=-9}^{8} u_j^{(1)} x^{j+18} + \sum_{j=-7}^{8} u_j^{(1)} x^j,
\]
\[
P_2^{(1)} = \sum_{j=-9}^{8} u_j^{(1)} x^{j+7},
\]
\[
P_3^{(1)} = u_9^{(1)} x^9.
\]

In order to compute \( S_2 \mod F(x) \), we first compute the polynomial \( G'_1 = G'_1 x^{-9} = \sum_{j=-9}^{9} u_j^{(1)} x^j \) as in [18, Section III-B] (see the middle part of Fig. 1 (a)). The computations of 6 polynomials

\[
A_0 + A_1 = \sum_{i=0}^{4} s_i x^i, \quad B_0 + B_1 = \sum_{i=0}^{4} t_i x^i,
\]
\[
A_0 + A_2 = \sum_{i=0}^{4} u_i x^i, \quad B_0 + B_2 = \sum_{i=0}^{4} v_i x^i,
\]
\[
A_1 + A_2 = \sum_{i=0}^{4} w_i x^i, \quad B_1 + B_2 = \sum_{i=0}^{4} z_i x^i.
\]

need 30 XOR gates and \( T_X \) delay. Since \( E_{u,v} = (A_u + A_v)(B_u + B_v) \) for \( 0 \leq u < v \leq 2 \), the product \( E_{0,1} = (A_0 + A_1)(B_0 + B_1) \) is obtained by adding the entries of the same row in the following matrix

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

Then, the coefficients of the polynomial \( G'_1 = E_{0,1} + E_{0,2}x^5 + E_{1,2}x^{10} \) can be obtained by adding the entries of the same row in the matrix \( E_{G'_1} \), using a binary XOR tree, where

\[
E_{G'_1} =
\]

The computation of the matrix \( E_{G'_1} \) costs 75 AND gates and \( T_A \) delay. The additions of the entries of the same row in \( E_{G'_1} \) require 56 XOR gates and 3\( T_X \) delays. (These computations consist of AND blocks and BTX blocks similar to that of \( e_0, \cdots, e_3 \) in Fig. 1 (b).) Consequently, the total complexity for the polynomial \( G'_1 \) (i.e., \( G'_1 \)) is 75 AND gates, 30+56=86 XOR gates, and \( 3T_A + 4T_X \) delays, as in (4). Next, the summation of polynomials \( P_1^{(1)} + P_2^{(1)} + P_3^{(1)} \) is performed with 3 XOR gates.

As a result, the total space complexity for \( S_2 \mod F(x) = P_1^{(1)} + P_2^{(1)} + P_3^{(1)} \) is 75 AND gates and 89 XOR gates (see (12)). Its time complexity is considered in Example 3.
2) COMPUTATION OF $S_3 \mod F(x)$

Here, we perform the computation of $S_3 \mod F(x)$ in (5). Its architecture is described in Fig. 2. The polynomial

$$S_3 = \left( \sum_{i=0}^{n-1} A_i x^{im_0} \right) \overline{A} + \left( \sum_{i=0}^{n-1} A_i x^{im_0} \right) \overline{B} + \overline{AB} x^{nm_0 - 2k}$$

can be written as $S_3 = \sum_{i=m-2k-r}^{2m-2k-2} f_i x^i$ since the degrees of $A_i$ and $B_i$ (resp. $\overline{A}$ and $\overline{B}$) are less than $m_0$ (resp. $r$). We first compute the coefficients $f_i$'s and then implement the modular reduction $\sum_{i=m-2k-r}^{2m-2k-2} f_i x^i \mod F(x)$.

The computation of the coefficients $f_i$'s is performed by computing two polynomials $(\sum_{i=1}^{n-i} B_i x^{im_0})\overline{A}$ and $(\sum_{i=0}^{n-1} A_i x^{im_0})\overline{B} + \overline{AB} x^{nm_0}$ in parallel followed by addition of these two polynomials. First, we consider the computation of $(\sum_{i=0}^{n-1} A_i x^{im_0})\overline{B} + \overline{AB} x^{nm_0}$. We use the following notations:

- $r \overline{A}_i$ := the first $r$ columns of the matrix $\overline{A}$,
- $\overline{b}$ := the coefficient vector of $\overline{B}$ with $r$ components,
- $\overline{b}'$ := the vector $[\overline{b}', 0, \ldots, 0]^T$ with $m_0$ components.

The polynomial $A_i \overline{B}$ can be written as the matrix-vector product

$$\begin{bmatrix} A_{i,L} \\ A_{i,H} \end{bmatrix} \overline{b}'$$

where $m_0 \times m_0$ matrices $A_{i,L}$ and $A_{i,H}$ are defined in (2). Then the polynomial $(\sum_{i=0}^{n-1} A_i x^{im_0})\overline{B}$ is represented as the matrix-vector product

$$(\sum_{i=0}^{n-1} A_i x^{im_0})\overline{B} = \begin{bmatrix} A_{0,L} \\ A_{0,H} \\ A_{1,L} \\ \vdots \\ A_{n-2,L} \\ A_{n-2,H} \\ A_{n-1,L} \\ A_{n-1,H} \end{bmatrix} \begin{bmatrix} \overline{b}' \\ \overline{b}' \\ \overline{b}' \\ \overline{b}' \end{bmatrix}$$

since the last $(m_0 - r)$ components of $\overline{b}'$ are all zero. (On the right side of the matrix, we denote the first exponent of indeterminate $x$ for each line.) Also, we write the polynomial $\overline{AB}$ as the matrix-vector product

$$\overline{AB} = \begin{bmatrix} \overline{A}_L \\ \overline{A}_H \end{bmatrix} \overline{b}$$

where

$$\overline{A}_L = \begin{bmatrix} a_{nm_0+0} & 0 & \cdots & 0 \\ a_{nm_0+1} & a_{nm_0+0} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{nm_0+r-2} & \cdots & a_{nm_0+0} \\ 0 & 0 & \cdots & a_{nm_0+r-1} \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

are $r \times r$ triangular matrices. Since the last $(m_0 - r)$ rows of $\overline{A}_{n-1,H}$ are all zero rows, we obtain that

$$\begin{bmatrix} r \overline{A}_{0,L} \\ r \overline{A}_{0,H} \\ r \overline{A}_{1,L} \\ \vdots \\ r \overline{A}_{n-2,L} \\ r \overline{A}_{n-2,H} \\ r \overline{A}_{n-1,L} \\ r \overline{A}_{n-1,H} \end{bmatrix} \overline{b} = (r \overline{A}_{0,L} + r \overline{A}_{1,L}) \overline{b} + (r \overline{A}_{n-2,H} + r \overline{A}_{n-1,L}) \overline{b}$$

(The computations of matrices $(r \overline{A}_{1,H} + r \overline{A}_{1,L})$ and $(r \overline{A}_{n-1,H} + r \overline{A}_{n-1,L})$ are all free.) The computation of the above matrix-vector products consists of AND blocks and BTX blocks similar to that of $e_i$'s (see Fig. 1 (b)). Since $(r \overline{A}_{0,L}) \overline{b}$ and $\overline{A}_H$ are $r \times r$ triangular matrices, the computations of the above matrix-vector products for $(\sum_{i=0}^{n-1} A_i x^{im_0})\overline{B} + \overline{AB} x^{nm_0} \overline{B}$
In parallel with the computation of \( S - M \cdot Park \), the complexity: \( \sum_{i=0}^{n-1} A_i x^{i m_0} \), \( A_i x^{i m_0} \) is similarly computed with the complexity: \( \sum_{i=0}^{n-1} B_i x^{i m_0} \), \( \sum_{i=0}^{n-1} B_i x^{i m_0} \), \( \sum_{i=0}^{n-1} A_i x^{i m_0} \) and \( \sum_{i=0}^{n-1} B_i x^{i m_0} \) are obtained by adding two polynomials \( \sum_{i=0}^{n-1} A_i x^{i m_0} \), \( \sum_{i=0}^{n-1} B_i x^{i m_0} \), \( \sum_{i=0}^{n-1} A_i x^{i m_0} \) and \( \sum_{i=0}^{n-1} B_i x^{i m_0} \) with \( (nm_0 + r - 1) \) XOR gates and \( T_X \) delays. Next, the coefficients \( f_i \)'s are free since they consist of \( m \) XOR gates. (If \( m_0 \) is written as a summation of 3 terms \( (2 \text{ terms if } r = 1) \) with the delay \( T_A + (1 + [\log_2 m_0])T_X \) by (13), we can write \( S_3 \mod F(x) \) as a summation of \( w \) terms \( S_3 \mod F(x) = Q_1 + \cdots + Q_w \) (15) within \( T_A + (1 + [\log_2 m_0])T_X \), where \( w = \begin{cases} 1 & \text{if } r = 1 \text{ or } r \leq m_0, \\ \frac{m_0}{2} & \text{if } \frac{m_0}{2} < r \leq 2, \\ \frac{m_0}{2} & \text{if } \frac{m_0}{2} < r < m_0. \end{cases} \)

3) COMPLEXITY FOR \( S_2 + S_3 \mod F(x) \)

The space complexities for \( (S_2 \mod F(x)) \) and \( (S_3 \mod F(x)) \) are given in (12) and (14), respectively. Finally, the addition of \( (S_2 \mod F(x)) + (S_3 \mod F(x)) \) requires \( m(= nm_0 + r) \) XOR gates. Therefore, the total space complexity for \( S_2 + S_3 \mod F(x) \) is equal to

\[
\begin{align*}
\text{\# AND} &= \frac{1}{2} n^2 m_0^2 - \frac{1}{2} n m_0^2 + 2 nm_0 + r^2, \\
\text{\# XOR} &= \frac{1}{2} n^2 m_0^2 - \frac{1}{2} n m_0^2 + \frac{5}{4} n^2 m_0 + (2r - 2) nm_0 - \frac{1}{2} nk + (r + 1) \left( \frac{1}{2} k \right) + \max(m - k, r) + r^2 \\
&\quad + \left( 2 \left( \frac{1}{2} k \right) - k \right) (r + 1) - \frac{r + 1}{2} n \\
&\quad - \left( 2 \left( \frac{1}{2} k \right) + \frac{1}{4} k \right) + \frac{1}{4} m_0 + Y_2 \\
&\text{if } n : \text{ even, } \\
&\text{else: odd.}
\end{align*}
\]

(16)

Now, we consider the time complexity for the computation of

\[
(S_2 \mod F(x)) + (S_3 \mod F(x))
\]

\[
= \begin{cases} \\
\sum_{i=1}^{\frac{n}{2}} p_1^{(i)} + \sum_{i=1}^{\frac{n}{2}} p_2^{(i)} + P_3 \\
\sum_{i=1}^{\frac{n-1}{2}} p_1^{(i)} + \sum_{i=1}^{\frac{n-1}{2}} p_2^{(i)} + P_3 \\
\text{if } t : \text{ even, } k \leq \text{mod } 0 - r - 2, \\
0 \quad \text{otherwise}
\end{cases}
\]

(17)

from (11) and (15), where \( P_3 := \sum_{i=1}^{n-1} P_3^{(i)} \). According to Table 4, the computation of \( P_3 \) is free since it consists of non-overlapped parts. Each computation of \( P_1^{(i)} \), \( P_2^{(i)} \), \( P_3 \), and \( P_2' \) is implemented within the delay \( T_A + (1 + [\log_2 m_0])T_X \) since they are obtained without any costs from the polynomials \( G_i \)'s (see (4)). Also, the delay for each \( Q_i \) is \( T_A + (1 + [\log_2 m_0])T_X \) by (15). Thus, after \( T_A + (1 + [\log_2 m_0])T_X \) delays, \( (S_2 \mod F(x)) + (S_3 \mod F(x)) \) is written as a summation of at most

\[
\begin{align*}
2^n + w &= (n + w) \text{ terms if } n : \text{ even, } \\
2^n - 1 + 2w &= (n + w) \text{ terms if } n : \text{ odd.}
\end{align*}
\]
(Here, we note that if \( n \) is odd, \( t \) is even, \( k \leq tm_0 - r - 2 \), then the ranges of degrees of \( P_2(x) \) and \( P_3 \) are \([m_0 - k, -1] \) and \([0, m_0 - k - r - 2] \), respectively, from Table 4 and so their addition is free. Therefore, \( S_2 \) mod \( F(x) \) can be regarded as a summation of \((2^\frac{m_0}{2} + 1) \) terms in this case, too.) As a result, the total delay for \((S_2 \mod F(x)) + (S_3 \mod F(x))\) is

\[
T_A + (1 + \left[ \log_2 m_0 \right] + \left[ \log_2(n + w) \right]) TX.  \quad (18)
\]

**Example 3:** Here, we consider the computation of \( S_3 \mod F(x) \) and the time complexity for \( S_2 + S_3 \mod F(x) \) in Examples 1 and 2.

We first compute the polynomial

\[
S_3 = \left( \frac{1}{2} \sum_{i=0}^{2} B_i x^{S_i} \right) \bar{A} + \left( \frac{1}{2} \sum_{i=0}^{2} A_i x^{S_i} \right) \hat{B} + \bar{A} \hat{B} x^{15} = \sum_{i=0}^{20} f_i x^i.
\]

The polynomial \((\sum_{i=0}^{2} A_i x^{S_i}) \hat{B} + \bar{A} \hat{B} x^{15}\) can be represented as the matrix-vector product

\[
(\sum_{i=0}^{2} A_i x^{S_i}) \bar{B} + \bar{A} \hat{B} x^{15} = \left[ \begin{array}{c} 3A_{0,L} \bar{b} \\ 3A_{1,L} \bar{b} \\ 3A_{2,L} \bar{b} \end{array} \right],
\]

where

\[
3A_{i,L} = \begin{bmatrix} a_{S_i} & 0 & 0 \\ a_{S_i+1} & a_{S_i} & 0 \\ a_{S_i+2} & a_{S_i+1} & a_{S_i} \\ a_{S_i+3} & a_{S_i+2} & a_{S_i+1} \\ a_{S_i+4} & a_{S_i+3} & a_{S_i+2} \end{bmatrix}, \quad \bar{A}_L = \begin{bmatrix} a_{15} & 0 & 0 \\ a_{16} & a_{15} & 0 \\ a_{17} & a_{16} & a_{15} \end{bmatrix},
\]

\[
3A_{i,H} = \begin{bmatrix} 0 & a_{S_i+4} & a_{S_i+3} \\ 0 & 0 & a_{S_i+4} \end{bmatrix}, \quad \bar{A}_H = \begin{bmatrix} 0 & a_{17} & 0 \\ 0 & 0 & a_{16} \\ 0 & 0 & 0 \end{bmatrix},
\]

\[
\bar{b} = [b_{15}, b_{16}, b_{17}, 1]^T.
\]

The above matrix-vector product can be computed with 54 AND gates, 34 XOR gates, and \( T_A + 2T_X \) delays. In parallel with the computation of \((\sum_{i=0}^{2} B_i x^{S_i}) \bar{A} + \bar{A} \hat{B} x^{15} \), the polynomial \((\sum_{i=0}^{2} B_i x^{S_i}) \bar{A} + \bar{A} \hat{B} x^{15} \) is similarly computed with 45 AND gates, 28 XOR gates, and \( T_A + 2T_X \) delays. And then, we get the polynomial \( S_3 = \sum_{i=1}^{20} f_i x^i \) by adding two polynomials \((\sum_{i=0}^{2} B_i x^{S_i}) \bar{A} + \bar{A} \hat{B} x^{15} \) and \((\sum_{i=0}^{2} A_i x^{S_i}) \bar{B} + \bar{A} \hat{B} x^{15} \) with 17 XOR gates and \( T_X \) delay. Therefore, the polynomial \( S_3 = \sum_{i=1}^{20} f_i x^i \) is computed with the complexity: 54+45=99 AND gates, 34+28+17=79 XOR gates, and \( T_A + 3T_X \) delays (see (13)).

Next, the modular reduction \( S_3 \mod F(x) = \sum_{i=1}^{20} f_i x^i \) mod \( F(x) \) is equal to

\[
\sum_{i=1}^{20} f_i x^i \mod F(x) = D_1 + D_2 + D_3,
\]

where

\[
D_1 = \sum_{i=1}^{10} f_i x^i + \sum_{i=11}^{18} f_i x^{i-18},
\]

\[
D_2 = \sum_{i=11}^{20} f_i x^{i-11}, \quad D_3 = \sum_{i=19}^{20} f_i x^{i-18}.
\]

The addition \( D_1 + D_2 + D_3 \) requires 12 XOR gates. Hence, the total space complexity for \( S_3 \mod F(x) \) is \( 99 \) AND gates and \( 79+12=91 \) XOR gates (see (14)). Since \( \frac{m_0}{2} < r, w = 3 \) and \( Q_1 = D_1 \) in (15) for \( 1 \leq i \leq 2 \).

Now, we consider the total time complexity for the computation of \( S_2 + S_3 \mod F(x) \). According to the above description and Example 2, we can write

\[
(S_2 \mod F(x)) + (S_3 \mod F(x)) = P_1^{(1)} + P_2^{(1)} + P_3^{(1)} + Q_1 + Q_2 + Q_3,
\]

where \( P_i^{(1)} \) and \( Q_i \) can be computed in parallel within the delay \( T_A + 4T_X \). The above summation is implemented with \( 3T_X \) delays. Thus, the total time complexity for \( S_2 + S_3 \mod F(x) \) is \( T_A + 7T_X \) delays (see (18)).

### C. OVERALL COMPLEXITY FOR FIELD MULTIPLICATION C

We implement the computations \((S_1 \mod F(x)) \) and \((S_2+S_3 \mod F(x)) \) in parallel with \( T_A + (1 + \left[ \log_2 m_0 \right] + \left[ \log_2(n + w) \right]) TX \). Then, the addition of these two polynomials is performed with \( m(n = m_0 + r) \) XOR gates and \( T_X \) delay. As the final outcome, the overall complexity for the field multiplication \( C = (S_1 \mod F(x)) + (S_2 + S_3 \mod F(x)) \) is equal to

\[
\text{Delay} = T_A + (2 + \left[ \log_2 m_0 \right] + \left[ \log_2(n + w) \right]) TX,
\]

\[
\# \text{AND} = \frac{1}{2} n^2 m_0^2 + \frac{1}{2} n m_0^2 + 2nm_0 + r^2,
\]

\[
\# \text{XOR} = \frac{1}{2} n^2 m_0^2 + \frac{1}{2} n^2 m_0^2 + 3 n^2 m_0 + (2r - \frac{3}{2}) nm_0,
\]

\[
- \frac{1}{2} nk + rW(n - 1) + (r + 1) f_2 + J + r^2 - r + \max\{m - 2r, k\} + \max\{m - k, r\}
\]

\[
+ \left\{ \begin{array}{ll}
2f_1^2 + \frac{1}{2} nk + \frac{1}{2} m - \frac{1}{2} l^2 m_0 + Y_1 & \text{if } n : \text{ even} \\
2f_1^2 + \frac{1}{2} nk + \frac{1}{2} m + \frac{1}{2} l^2 m_0 + Y_2 & \text{if } n : \text{ odd}
\end{array} \right.
\]

using (10), (16), and (18). Since \( \sum_{i=1}^{l} W(i) \) can be roughly written as \( \frac{1}{2} \log_2 l \) for a nonzero integer \( l \) ([22]), \( J \) defined in Table 3 is roughly estimated as \( O(m \log_2 n) \). The above number of XOR gates for the computation of the field multiplication can be written as

\[
\frac{1}{2} m^2 + \frac{1}{2} m_0 m + \frac{3}{2} nm - \frac{1}{2} nk + tk - \frac{1}{2} l^2 m_0
\]

\[
+ rm - \frac{1}{2} r^2 m_0 - \frac{1}{2} r^2 + O(m \log_2 n)
\]

by omitting the linear parts. We note that \( rm - \frac{1}{2} r^2 m_0 - \frac{1}{2} r^2 < m_0 m \) and \( -\frac{1}{2} nk + tk - \frac{1}{2} l^2 m_0 \leq -\frac{1}{2} nk + tk - \frac{1}{2} l^2 k \leq \frac{1}{2} k \) since \( 1 \leq t \leq n + 1 \).

### IV. COMPARISON

In Table 5, we compare the proposed multiplier with the best-known bit-parallel multipliers for \( GF(2^m) \) defined by a...
### TABLE 5. Comparison of the best-known bit-parallel multipliers for trinomial $x^m + x^k + 1$.

| Multiplier | $m > 2k$ | $m = 2k$ | $m < 2k$ | $m = n m_0$ | $m = n m_0 + 1$ | $m = n m_0 + r$ | Delay |
|------------|----------|----------|----------|-------------|----------------|-----------------|-------|
|            | $m^2$    | $m^2$ $\times$ | $m^2$ $\times$ | $\frac{1}{2} m^2 + \frac{1}{2} m m_0$ | $(m - 1) m_0 - \frac{1}{2}$ | $\frac{1}{2} m^2 + \frac{1}{2} m m_0 + \frac{1}{2} n m_0$ | $\frac{1}{2} m^2 + \frac{1}{2} m m_0 + \frac{1}{2} n m_0$ | $\frac{1}{2} m^2 + \frac{1}{2} m m_0 + \frac{1}{2} n m_0$ |
| [8], [11]  | $T_A + [\log_2 (2m - k - 1)] T_X$ | $T_A + [\log_2 (3m/2)] T_X$ | $T_A + [\log_2 (m + k)] T_X$ | $T_A + (2 + [\log_2 m_0] + [\log_2 (n + 1)]) T_X$ | $T_A + (2 + [\log_2 m_0] + [\log_2 (n + 1)]) T_X$ | $T_A + (2 + [\log_2 m_0] + [\log_2 (n + 1)]) T_X$ | $T_A + (2 + [\log_2 m_0] + [\log_2 (n + 1)]) T_X$ | $T_A + (2 + [\log_2 m_0] + [\log_2 (n + 1)]) T_X$ |
| [18]       | $(1 < n, m_0)$ | $(1 < n, m_0)$ | $(1 < n, m_0)$ | $0 < r < m_0$ | $0 < r < m_0$ | $0 < r < m_0$ | $0 < r < m_0$ | $0 < r < m_0$ |
|            | $T_A + \frac{1}{2} m m_0 O(m m_0)$ | $T_A + \frac{1}{2} m m_0 O(m m_0)$ | $T_A + \frac{1}{2} m m_0 O(m m_0)$ | $T_A + \frac{1}{2} m m_0 O(m m_0)$ | $T_A + \frac{1}{2} m m_0 O(m m_0)$ | $T_A + \frac{1}{2} m m_0 O(m m_0)$ | $T_A + \frac{1}{2} m m_0 O(m m_0)$ | $T_A + \frac{1}{2} m m_0 O(m m_0)$ |
| This paper  | $m = n m_0 + r$ | $m = n m_0 + r$ | $m = n m_0 + r$ | $m = n m_0 + r$ | $m = n m_0 + r$ | $m = n m_0 + r$ | $m = n m_0 + r$ | $m = n m_0 + r$ |
|            | $0 < r < m_0$, $1 < n$ | $0 < r < m_0$, $1 < n$ | $0 < r < m_0$, $1 < n$ | $0 < r < m_0$, $1 < n$ | $0 < r < m_0$, $1 < n$ | $0 < r < m_0$, $1 < n$ | $0 < r < m_0$, $1 < n$ | $0 < r < m_0$, $1 < n$ |
|            | $-\frac{1}{2} r^2$ | $-\frac{1}{2} r^2$ | $-\frac{1}{2} r^2$ | $-\frac{1}{2} r^2$ | $-\frac{1}{2} r^2$ | $-\frac{1}{2} r^2$ | $-\frac{1}{2} r^2$ | $-\frac{1}{2} r^2$ |

### TABLE 6. Complexity comparison of multipliers for some fields defined by trinomial $x^m + x^k + 1$, where $m = n m_0 + r$.

| $m$, $n$, $r$, $d$, $e$ | # AND | # XOR | Delay |
|--------------------------|-------|-------|-------|
| 233                      | 54289 | 54288 | $T_A + [\log_2 (2m - k - 1)] T_X$ |
| [18]                     | 23005 | 38676 | $T_A + [\log_2 (3m/2)] T_X$ |
| [19]                     | 30741 | 33983 | $T_A + [\log_2 (m + k)] T_X$ |
| this paper               | 28305 | 38676 | $T_A + [\log_2 (m + k)] T_X$ |
| 409                      | 167281 | 109820 | $T_A + [\log_2 (m + k)] T_X$ |
| [8]                      | 83684 | 117268 | $T_A + [\log_2 (m + k)] T_X$ |
| [18]                     | 88945 | 100290 | $T_A + [\log_2 (m + k)] T_X$ |
| [19]                     | 90450 | 109812 | $T_A + [\log_2 (m + k)] T_X$ |
| this paper               | 90750 | 100148 | $T_A + [\log_2 (m + k)] T_X$ |
| 415                      | 172225 | 172224 | $T_A + [\log_2 (m + k)] T_X$ |
| [18]                     | 186941 | 208344 | $T_A + [\log_2 (m + k)] T_X$ |
| [19]                     | 191288 | 101695 | $T_A + [\log_2 (m + k)] T_X$ |
| this paper               | 252249 | 141716 | $T_A + [\log_2 (m + k)] T_X$ |
| 431                      | 178929 | 178928 | $T_A + [\log_2 (m + k)] T_X$ |
| [18]                     | 178929 | 178928 | $T_A + [\log_2 (m + k)] T_X$ |
| [19]                     | 178929 | 178928 | $T_A + [\log_2 (m + k)] T_X$ |
| this paper               | 178929 | 178928 | $T_A + [\log_2 (m + k)] T_X$ |
| 439                      | 105761 | 105760 | $T_A + [\log_2 (m + k)] T_X$ |
| [18]                     | 105761 | 105760 | $T_A + [\log_2 (m + k)] T_X$ |
| [19]                     | 105761 | 105760 | $T_A + [\log_2 (m + k)] T_X$ |
| this paper               | 105761 | 105760 | $T_A + [\log_2 (m + k)] T_X$ |
| 447                      | 192721 | 192720 | $T_A + [\log_2 (m + k)] T_X$ |
| [18]                     | 192721 | 192720 | $T_A + [\log_2 (m + k)] T_X$ |
| [19]                     | 192721 | 192720 | $T_A + [\log_2 (m + k)] T_X$ |
| this paper               | 192721 | 192720 | $T_A + [\log_2 (m + k)] T_X$ |

The fastest bit-parallel multipliers for trinomials have been proposed in [8] and [11]. [18] proposed multipliers for trinomial $x^m + x^k + 1$ based on $n$-term KA, where $m$ is decomposed as $m = nm_0$ or $m = nm_0 + 1$. 

VOLUME 8, 2020 173505
In [19], such limitation for \( m \) in [18] is loosened, that is, \( m \) satisfies \( m = nm_0 + r \) with \( 0 \leq r < n, m_0 \), but there is the limitation \( m \geq 2k \) for trinomials. The multiplier in [18] or [19] achieves the lowest space complexity among similar bit-parallel multipliers.

The proposed multiplier in this paper further eases limitations for trinomials and a decomposition of \( m \) in [18], [19]. In fact, the multiplier for trinomial \( x^{nm_0+1} + x^{k} + 1 \) in [18] can be considered as a special case of the proposed multiplier. The complexity of the proposed multiplier is comparable with those in [18] and [19]. It is difficult to immediately know which multiplier is better. However, more flexible options for trinomials and values for \( n, m_0 \) and \( r \) may yield better complexities. In Table 6, we prove this by giving specific comparisons for several fields which are addressed in [19].

Compared to [18] and [19], the proposed multiplier has lower space complexity while having the same time complexity as shown in Table 6. In some cases such as \( m = 439 \) or 447, the proposed multiplier has not only lower space complexity but also lower time complexity.

Let us compare the proposed multiplier with the fastest multiplier in [8]. The proposed multiplier has one more \( T_X \) delay. However, its space complexity is much lower. For instance, when \( m = 439 \), the proposed multiplier has about 43% reduced space complexity according to Table 6. Our space complexity gain (about 43% reduced space complexity) is much greater than the time complexity loss (about 10% XOR gate delay increment).

V. CONCLUSION

We have proposed a new hybrid multiplier for trinomial \( x^{nm_0+r} + x^{k} + 1 \) based \( n \)-term KA, which is a generalization of the multiplication scheme for \( x^{nm_0+1} + x^{k} + 1 \) in [18]. We have evaluated the explicit complexity of the proposed multiplier and compared it with the best-known bit-parallel multipliers for trinomials. The specific comparisons for several fields show that the proposed multiplier achieves the lowest space complexity with the same or lower time complexity among hybrid multipliers. Compared to the fastest multipliers, the time complexity of the proposed multiplier costs \( T_X \) higher while its space complexity is much lower. It is noted that the space complexity gain of the proposed multiplier is much greater than its time complexity loss. Consequently, the proposed multiplier outperforms similar bit-parallel multipliers.

REFERENCES

[1] R. Lidl and H. Niederreiter, Introduction to Finite Fields and Their Applications, New York, NY, USA: Cambridge Univ. Press, 1994.

[2] R. Lidl and H. Niederreiter, Finite Fields, New York, NY, USA: Cambridge Univ. Press, 1996.

[3] Digital Signature Standard, Federal Information Processing Standards Publication, Standard 186-2, National Institute Standards Technology, 2000.

[4] B. Sunar, “A generalized method for constructing subquadratic complexity \( GF(2^n) \) multipliers,” IEEE Trans. Comput., vol. 53, no. 9, pp. 1097–1105, Sep. 2004.

[5] H. Fan, J.-G. Sun, M. Gu, and K.-Y. Lam, “Overlap-free Karatsubafornan polynomial multiplication algorithms,” IET Inf. Secur., vol. 4, no. 1, pp. 8–14, Mar. 2010.

[6] 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, May 2015.

[7] 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, Apr. 2005.

[8] H. Fan and M. A. Hasan, “Fast bit parallel-shifted polynomial basis multipliers in \( GF(2^n) \),” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 12, pp. 2606–2615, Dec. 2006.

[9] E. D. Mastrovito, “VLSI architectures for computation in Galois fields,” Ph.D. dissertation, Dept. Elect. Eng., Linköping Univ., Linköping, Sweden, 1991.

[10] A. Reyhani-Masoleh and M. A. Hasan, “Low complexity bit parallel architectures for polynomial basis multiplication over \( GF(2^m) \),” IEEE Trans. Comput., vol. 53, no. 8, pp. 945–959, Aug. 2004.

[11] A. Hariri and A. Reyhani-Masoleh, “Bit-serial and bit-parallel Montgomery multiplication and squaring over \( GF(2^n) \),” IEEE Trans. Comput., vol. 58, no. 10, pp. 1332–1345, Oct. 2009.

[12] A. Weimerskirch and C. Paar, “Generalizations of the Karatsuba algorithm for efficient implementations,” Cryptol. ePrint Arch., Tech. Rep. 2006/224, 2006. [Online]. Available: http://eprint.iacr.org.

[13] M. Elia, M. Leone, and C. Visentin, “Low complexity bit-parallel multipliers for \( GF(2^{2m}) \) with generator polynomial \( x^m + x + 1 \),” Electron. Lett., vol. 35, no. 7, pp. 551–552, 1999.

[14] Y. Li, G.-L. Chen, and J.-H. Li, “Speedup of bit-parallel Karatsuba multiplier in generated by trinomials,” Inf. Process. Lett., vol. 111, no. 8, pp. 390–394, Mar. 2011.

[15] H. Shen and Y. Jin, “Low complexity bit parallel multiplier for generated by equally-spaced trinomials,” Inf. Process. Lett., vol. 107, no. 6, pp. 211–215, Aug. 2008.

[16] 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, Sep. 2017.

[17] 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, 2018.

[18] S.-M. Park, K.-Y. Chang, D. Hong, and C. Seo, “Low space complexity \( GF(2^n) \) multiplier for trinomials using \( n \)-term Karatsuba algorithm,” IEEE Access, vol. 7, pp. 27047–27064, 2019.

[19] Y. Li, S. Sharma, Y. Zhang, X. Ma, and C. Qi, “On the complexity of hybrid \( n \)-term Karatsuba multiplier for trinomials,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 3, pp. 852–865, Mar. 2020.

[20] X.-N. Xie, G.-L. Chen, and Y. Li, “Novel bit-parallel multiplier for defined by all-one polynomial using generalized Karatsuba algorithm,” Inf. Process. Lett., vol. 114, no. 3, pp. 140–146, Mar. 2014.

[21] S.-M. Park, K.-Y. Chang, D. Hong, and C. Seo, “Space efficient \( GF(2^n) \) multiplier for special pentanomials based on \( n \)-term Karatsuba algorithm,” IEEE Access, vol. 8, pp. 27342–27360, 2020.

[22] Y. Li and Y. Chen, “New bit-parallel Montgomery multiplier for trinomials using squaring operation,” Integration, vol. 52, pp. 142–155, Jan. 2016.
S.-M. Park et al.: Efficient Bit-Parallel Multiplier for All Trinomials Based on n-Term Karatsuba Algorithm

KU-YOUNG CHANG received the B.S., M.S., and Ph.D. degrees in mathematics from Korea University, Seoul, South Korea, in 1995, 1997, and 2000. He is currently a Principal Researcher with the Cyber Security Research Division, Electronics and Telecommunication Research Institute, South Korea, where his research interests include applied cryptography and finite field theory.

DOWON HONG received the B.S., M.S., and Ph.D. degrees in mathematics from Korea University, Seoul, South Korea, in 1994, 1996, and 2000, respectively. He was a Principal Member of Engineering Staff with the Electronics and Telecommunications Research Institute, South Korea, from 2000 to 2012. He joined the Department of Applied Mathematics at Kongju National University, South Korea, in 2012. He has been a Full Professor since 2015. His research interests include cryptography and information security, data privacy, and digital forensics.

CHANGHO SEO received the B.S. and M.S. degrees from Korea University, South Korea, in 1990 and 1992, respectively, and the Ph.D. degree from the Department of Mathematics, Korea University, in 1996. He is currently a Full Professor with the Department of Applied Mathematics, Kongju National University, South Korea. His research interests include cryptography, information security, and system security.

***

VOLUME 8, 2020

173507