# Efficient Subquadratic Space Complexity Binary Polynomial Multipliers Based On Block Recombination

Murat Cenk, Anwar Hasan, Christophe Negre

# Efficient Subquadratic Space Complexity Binary Polynomial Multipliers Based On Block Recombination

**Abstract**

Some applications like cryptography involve a large number of multiplications of binary polynomial. In this paper we consider two, three and four-way methods for parallel implementation of binary polynomial multiplication. We propose optimized three and four-way split formulas which reduce the space and time complexity of the best known methods. Moreover, we present a block recombination method which provides some further reduction in the space complexity of the considered two, three and four-way split multipliers.

**Index Terms**

Binary polynomial multiplication, two-way split formula, three-way split formula, subquadratic space complexity, binary field, block recombination.

## 1 Introduction

Finite fields are part of important applications like coding theory and cryptography. For example, protocols over elliptic curves [7], [6] require several hundreds of multiplications and additions in a finite field of size $[2^{160}, 2^{600}]$. Efficient finite field arithmetic, specifically field multiplication and addition, is thus essential to obtain improved implementations of cryptographic protocols.

In this paper, we focus on bit parallel hardware implementation of multiplication over extended binary fields $\mathbb{F}_{2^n}$. Such a field can be defined as the set of binary polynomials modulo an irreducible polynomial $P$ of degree $n$. An addition of two elements in $\mathbb{F}_{2^n}$ consists of pairwise modulo-two addition of the coefficients, and this can be easily implemented with $n$ parallel XOR gates. The multiplication in $\mathbb{F}_{2^n}$ consists of a regular binary polynomial multiplication followed by a reduction modulo an irreducible polynomial $P$. When $P$ is a sparse polynomial, i.e., a trinomial or a pentanomial, the reduction consists of few shifts and additions and is thus quite simple to implement. The multiplication of polynomials is generally more costly. For the considered range of $n$, the most efficient methods to multiply two binary polynomials are the subquadratic method based on Karatsuba [8] or three-way split formulas [9] which have a complexity of $O(n^{1+\varepsilon})$ bit operations with $0 < \varepsilon < 1$.

During the past few years improvements have been made on the Karatsuba and three-way split formulas. Specifically, Bernstein has improved the space complexity of Karatsuba formula by optimizing its reconstruction part of the formula [1]. He has also extended this approach to two recursions of the Karatsuba formula, leading to a four-way split formula with a better space requirement than the original Karatsuba formula. In [2], Cenk *et al.* have applied the technique of Bernstein to improve the space complexity of the three-way split formula with six recursive multiplications. Finally, Fan *et al.* [3] have optimized the delay of two and three-way split formulas by using a different kind of splitting of the polynomials.

In this paper we propose some further optimizations of three and four-way formulas. Concerning the three-way split, we remove some redundant computation in the formula of [2] to reduce the space complexity. We then combine this idea with the approach of [3] to obtain a three-way split formula which has the same delay as the one in [3] but with a smaller space complexity. For the four-way formula, we combine the four-way approach of [1] and the modified splitting of [3], this yields a formula which has the same delay as [3] but with a lower space complexity.

In this paper, we also investigate a block recombination approach for the parallel multiplier based on two and three-way split formulas. The work presented here is an extension of the block recombination approach proposed in [5] for the Toeplitz matrix-vector product. We first state the basic recombination operation which reduces the space complexity of an architecture performing two multiplications in parallel followed by an addition. We then apply this recombination to different Karatsuba and three-way split multipliers. This results in multipliers with smaller space complexities while having the same delay.

The remainder of this paper is organized as follows: in Section 2 we review the best known two and three-way split formulas for binary polynomial multiplication. In Section 3 we propose some optimizations for three-way and four-way split formulas. We then present in Section 4 a block recombination approach for polynomial multiplication. We finally compare the complexity of our proposed methods to the best known methods and give some concluding remarks in Section 5.

## 2 Review of subquadratic methods for polynomial multiplications

Let us consider two binary polynomials $A = \sum_{i=0}^{n-1} a_i X^i$ and $B = \sum_{i=0}^{n-1} b_i X^i$ of degree $n-1$. The product $C = A \times B$ can be computed by a direct expansion of their respective expression

$$C = A \times B = \sum_{k=0}^{2n-2} \left( \sum_{0 \leq i,j < n, i+j=k} a_i b_j \right) X^k.$$

This method, known as the schoolbook method, requires $n^2$ bit multiplications and $(n-1)^2$ bit additions. When $n \in [160, 600]$ and the bit operations are performed in parallel fashion, this approach results in quite large circuit. For this size of $n$, subquadratic methods are generally more suitable for practical

applications. These methods split each polynomial in two or three and then express the product $C$ in terms of a number of products of smaller degrees. When this process is applied recursively, it results in $O(n^{\delta}),\ \delta \in \{\log_2(3), \log_3(6)\}$, bit operations for the whole multiplication. In this section we review the best known two and three-way split approaches.

### 2.1 Review of two-way formulas

Here we consider the two-way approach, also known as the Karatsuba method, for binary polynomial multiplication. We first review the original Karatsuba formula and then present some recent optimizations on the space and the time complexities.

#### *2.1.1 Original Karatsuba formula*

Polynomials $A(X)$ and $B(X)$ are in $\mathbb{F}_2[X]$. The original Karatsuba approach consists of splitting $A$ and $B$ in two parts

$$A(X) = \underbrace{\sum_{i=0}^{n/2-1} a_i X^i}_{A_0} + X^{n/2} \underbrace{\sum_{i=0}^{n/2-1} a_{i+n/2} X^i}_{A_1} \quad \text{and} \quad B(X) = \underbrace{\sum_{i=0}^{n/2-1} b_i X^i}_{B_0} + X^{n/2} \underbrace{\sum_{i=0}^{n/2-1} b_{i+n/2} X^i}_{B_1},$$

and then compute $C = A \times B$ in two steps:

- *Recursive product.* We perform three half size polynomial products

$$\begin{aligned} P_0 &= A_0 B_0, \\ P_1 &= A_1 B_1, \\ P_2 &= (A_0 + A_1)(B_0 + B_1). \end{aligned}$$

- *Reconstruction.* We reconstruct $C = A \times B$ as

$$C = P_0 + (P_0 + P_1 + P_2) X^{n/2} + P_1 X^n.$$

The three multiplications $P_0$, $P_1$ and $P_2$ are performed by applying the same method recursively. When each product $P_0$, $P_1$ and $P_2$ is performed in parallel and recursively, the resulting multiplier has the complexity given in (1) with a recursive form (in the left side) and a non-recursive form (in the right side). Note that $\mathcal{S}_{\oplus}(n)$ represents the number bit additions and $\mathcal{S}_{\otimes}(n)$ the number of bit multiplications and $D_{\oplus}$ and $D_{\otimes}$ represent the delay of a bit level addition and multiplication, respectively.

$$\left\{ \begin{array}{rcl} \mathcal{S}_{\oplus}(n) &=& 4n - 4 + 3\mathcal{S}_{\oplus}(n) \\ \mathcal{S}_{\otimes}(n) &=& 3\mathcal{S}_{\otimes}(n) \\ \mathcal{D}(n) &=& 3D_{\oplus} + \mathcal{D}(n/2) \end{array} \right. \qquad \left\{ \begin{array}{rcl} \mathcal{S}_{\oplus}(n) &=& 6n^{\log_2(3)} - 8n + 2 \\ \mathcal{S}_{\otimes}(n) &=& n^{\log_2(3)} \\ \mathcal{D}(n) &=& 3\log_2(n) D_{\oplus} + D_{\otimes} \end{array} \right. \tag{1}$$

#### 2.1.2 Bernstein's optimization of the Karatsuba reconstruction

Recently some improvements have been proposed for the complexity of the Karatsuba formula. In particular, Bernstein in [1] has reduced the complexity of the reconstruction. He has proposed to reconstruct $C$ in three steps as follows:

- *Step 1.* $R_0 = P_0 + X^{n/2}P_1$, which requires $n/2 - 1$ bit additions.
- *Step 2.* $R_1 = R_0(1 + X^{n/2})$, which requires $n - 1$ bit additions.
- *Step 3.* $C = R_1 + P_2X^{n/2}$, which requires $n - 1$ bit additions.

This optimization saves $n/2-1$ bit additions from the original Karatsuba reconstruction. When applied in parallel and recursively, this approach reduces the total number of bit additions to $\mathcal{S}_{\oplus}(n) = 5.5n^{\log_2(3)} - 7n + 3/2$ but requires a delay of $\mathcal{D}(n) = 3\log_2(n)D_{\oplus} + D_{\otimes}$.

**Remark 1.** The above formulas have been also presented by Zhou and Michalik in [10], independently from Bernstein,.

#### 2.1.3 Overlap free method for two-way split multiplication

Fan *et al.* in [3] have a different way to split the polynomials $A$ and $B$ in the Karatsuba formula leading to a reduction in the delay of the resulting parallel multiplier. Indeed, following [3], we separate the even and odd coefficients of $A$ and $B$ as follows

$$\begin{aligned} A &= \left(\sum_{i=0}^{n/2-1} a_{2i}X^{2i}\right) + X\left(\sum_{i=0}^{n/2-1} a_{2i+1}X^{2i}\right) = A_0(X^2) + XA_1(X^2), \\ B &= \left(\sum_{i=0}^{n/2-1} b_{2i}X^{2i}\right) + X\left(\sum_{i=0}^{n/2-1} b_{2i+1}X^{2i}\right) = B_0(X^2) + XB_1(X^2). \end{aligned}$$

The terms $A_i(X^2)$ and $B_i(X^2)$, for $i = 0, 1$, are considered as degree $n/2 - 1$ polynomial in $Y$ evaluated at $X^2$. The resulting overlap free two-way split formula is as follows:

- *Recursive product.* We perform three half size polynomial products

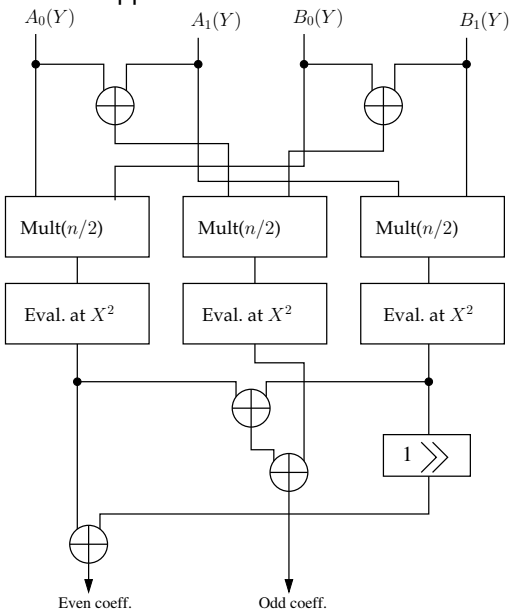
$$\begin{aligned} P_0(Y) &= A_0(Y)B_0(Y), \\ P_1(Y) &= A_1(Y)B_1(Y), \\ P_2(Y) &= (A_0(Y) + A_1(Y))(B_0(Y) + B_1(Y)). \end{aligned} \tag{2}$$

- *Reconstruction.* We then reconstruct $C$ as

$$C = P_0(X^2) + (P_0(X^2) + P_1(X^2) + P_2(X^2))X + P_1(X^2)X^2. \tag{3}$$

The space requirement ($\mathcal{S}_{\oplus}(n)$ and $\mathcal{S}_{\otimes}(n)$) is unchanged compared to the original Karatsuba formula (1), but the delay is reduced. In order to illustrate, in Fig 1, we show the data flow of the formulas (2) and (3). Since the blocks "Eval. at $X^2$" do not involve any gates, the critical path delay of Fig 1 is equal to $\mathcal{D}(n) = 2D_{\oplus} + \mathcal{D}(n/2)$. When applied recursively, this results in a delay of $\mathcal{D}(n) = 2\log_2(n)D_{\oplus} + D_{\otimes}$. This is about a 33% improvement in the delay complexity with respect to the original Karatsuba method.

Fig. 1. Data flow graph of the Fan *et al.* approach




### 2.2 Review of three-way multiplication

In this subsection, we review the best known three-way split formulas with six recursive multiplications. Specifically, we review the formula proposed in [2] which has the smallest space complexity and the formula of Fan *et al.* [3] which has the smallest delay among all formulas.

#### *2.2.1 Space optimized three-way split formulas*

As before $A(X)$ and $B(X)$ are two binary polynomials of degree $n-1$ in $\mathbb{F}_2[X]$ where $n$ is a power of 3. We split these two polynomials in three parts and replace $X^{n/3}$ by a new indeterminate $Y$. This results in two polynomials $A(Y) = A_0 + A_1Y + A_2Y^2$ and $B(Y) = B_0 + B_1Y + B_2Y^2$ of degree 2 in $Y$. The method given in [9] performs the multiplication $A(Y) \times B(Y)$ modulo $Y(Y+1)(Y^2+Y+1)(Y+\infty)$ and uses the Chinese remainder theorem to break this product into a number of smaller products: three products modulo a degree one term in $Y$ and one product modulo a degree 2 term in $Y$:

$$\begin{aligned}
C(0) &= A(0)B(0) = A_0B_0 \qquad \text{(Mult. mod } Y\text{)},\\
C(1) &= A(1)B(1) = (A_0+A_1+A_2)(B_0+B_1+B_2) \qquad \text{(Mult. mod } (Y+1)\text{)},\\
C'(Y) &= A(Y)B(Y) \mod (Y^2+Y+1)\\
&= (A_0+A_2+(A_1+A_2)Y)(B_0+B_2+(B_1+B_2)Y) \mod (Y^2+Y+1),\\
C(\infty) &= A(\infty)B(\infty) = A_2B_2 \qquad \text{(Mult. mod } (Y+\infty)\text{)}.
\end{aligned}$$

The multiplication modulo $(Y^2+Y+1)$ consists of a product of two degree one polynomials in $Y$. To perform this product we can use the Karatsuba formula, which requires three multiplications $(A_0+A_2)(B_0+B_2)$ and $(A_0+A_1)(B_0+B_1)$ and $(A_1+A_2)(B_1+B_2)$. We reconstruct $C$ from $C(0)$, $C(1)$, $C'(Y)$

and $C(\infty)$ using the Chinese remainder theorem. In Table 1 we have reported the formula of [2] which is an optimized version of this three-way split formula. Indeed, in Table 1, the three terms $P_3, P_4$ and $P_5$ are the multiplications corresponding to the product modulo $Y^2+Y+1$, $P_0$ and $P_2$ correspond to $C(0)$ and $C(\infty)$, respectively. A simpler product $P_1 = A_1B_1$ is used in place of $C(1)$.

TABLE 1

Three-way split formula with six multiplications [2]

| Operations | Computations | Cost | |
|---|---|---|---|
| | | $\#\oplus$ | $\#\otimes$ |
| Rec. Prod. | $P_0 = A_0B_0$ | $S_\oplus(n/3)$ | $S_\otimes(n/3)$ |
| | $P_1 = A_1B_1$ | $S_\oplus(n/3)$ | $S_\otimes(n/3)$ |
| | $P_2 = A_2B_2$ | $S_\oplus(n/3)$ | $S_\otimes(n/3)$ |
| | $P_3 = (A_0+A_1)(B_0+B_1)$ | $S_\oplus(n/3)+2n/3$ | $S_\otimes(n/3)$ |
| | $P_4 = (A_0+A_2)(B_0+B_2)$ | $S_\oplus(n/3)+2n/3$ | $S_\otimes(n/3)$ |
| | $P_5 = (A_1+A_2)(B_1+B_2)$ | $S_\oplus(n/3)+2n/3$ | $S_\otimes(n/3)$ |
| Reconst. | $R_0 = (P_0 + X^{n/3}P_1 + X^{2n/3}P_2)$ | $(2n/3-2)$ | 0 |
| | $R_1 = R_0(1+X^{n/3}+X^{2n/3})$ | $(2n-2)$ | 0 |
| | $C = R_1 + P_3X^{n/3} + P_4X^{2n/3} + P_5X^{3n/3}$ | $(2n-3)$ | 0 |
| Total | | $6\mathcal{S}_\oplus(n/3)+20n/3-7$ | $6\mathcal{S}_\otimes(n/3)$ |

We have also reported in Table 1 the number of bit additions ($\#\oplus$) and bit multiplications ($\#\otimes$) of each step of the three-way split formula. The resulting overall cost, in terms of bit addition $S_\oplus(n)$ and bit multiplication $S_\otimes(n)$, has the recursive form $S_\oplus(n) = 6\mathcal{S}_\oplus(n/3) + 20n/3 - 7$ and $\mathcal{S}_\otimes(n) = 6\mathcal{S}_\otimes(n/3)$. Solving these equations with the initial condition $\mathcal{S}_\oplus(1) = 0$ and $\mathcal{S}_\otimes(1) = 1$ gives

$$\begin{cases} \mathcal{S}_\oplus(n) &= \frac{79}{15}n^{\log_3(6)} - 20n/3 + 7/5, \\ \mathcal{S}_\otimes(n) &= n^{\log_3(6)}. \end{cases} \tag{4}$$

Now, if we assume that the computations are performed in parallel, the delay of this formula is equal to $4D_\oplus + \mathcal{D}(n/3)$, where $D_\oplus$ represents the delay of a bit addition and $\mathcal{D}(n/3)$ the delay of the product of two degree $n/3$ polynomials. We then can transform this recursive expression to a non-recursive form

$$\mathcal{D}(n) = 4\log_3(n)D_\oplus + D_\otimes.$$

#### 2.2.2 Delay reduced three-way split formula

We now review the three-way split formula proposed by Fan *et al.* in [3] which reduces the delay. Specifically, Fan *et al.* in [3] have proposed to split the polynomials $A$ and $B$ as follows:

$$\begin{aligned} A &= \left(\sum_{i=0}^{n/3} a_{3i}X^{3i}\right) + X\left(\sum_{i=0}^{n/2} a_{3i+1}X^{3i}\right) + X^2\left(\sum_{i=0}^{n/2} a_{3i+2}X^{3i}\right) = A_0(X^3) + XA_1(X^3) + X^2A_2(X^3), \\ B &= \left(\sum_{i=0}^{n/3} b_{3i}X^{3i}\right) + X\left(\sum_{i=0}^{n/2} b_{3i+1}X^{3i}\right) + X^2\left(\sum_{i=0}^{n/2} b_{3i+2}X^{3i}\right) = B_0(X^3) + XB_1(X^3) + X^2B_2(X^3). \end{aligned}$$

We can view $A_i(X^3)$ and $B_i(X^3), i = 0, 1, 2$, as degree $n/3 - 1$ polynomials in $Y$ evaluated at $X^3$. Now we consider $A = A_0(Y) + A_1(Y)X + A_2(Y)X^2$ and $B = B_0(Y) + B_1(Y)X + B_2(Y)X^2$ as polynomials of degree 2 in $X$. Following Fan *et al.* [3], we apply the three-way split formula as given below

- *Recursive product.* We perform the following six products of polynomial of size $n/3$

$$\begin{aligned} P_0(Y) &= A_0(Y)B_0(Y), \\ P_1(Y) &= A_1(Y)B_1(Y), \\ P_2(Y) &= A_2(Y)B_2(Y), \\ P_3(Y) &= (A_0(Y) + A_1(Y))(B_0(Y) + B_1(Y)), \\ P_4(Y) &= (A_0(Y) + A_2(Y))(B_0(Y) + B_2(Y)), \\ P_5(Y) &= (A_1(Y) + A_2(Y))(B_1(Y) + B_2(Y)), \end{aligned}$$

- *Reconstruction.* We then replace $Y$ by $X^3$ in $P_i$ for $i = 0, \ldots, 5$, and reconstruct $C = A \times B$ as follows

$$\begin{aligned} C &= \left( \overbrace{P_0(X^3) + X^3\left(P_5(X^3) + P_1(X^3) + P_2(X^3)\right)}^{C_0} \right) + X \left( \overbrace{P_3(X^3) + P_0(X^3) + P_1(X^3) + X^3 P_2(X^3)}^{C_1} \right) \\ &\quad + X^2 \left( \underbrace{P_4(X^3) + P_0(X^3) + P_1(X^3) + P_2(X^3)}_{C_2} \right). \end{aligned}$$

The computation in $C_0$, $C_1$ and $C_2$ are independent. The authors in [3] have determined the complexity of the above formula by separately evaluating the complexity of the three terms $C_0$, $C_1$ and $C_2$, and have obtained the following recursive and non-recursive form of the complexities:

$$\begin{cases} \mathcal{S}_\oplus(n) &= 6\mathcal{S}_\oplus(n/3) + \frac{22n}{3} - 10, \\ \mathcal{S}_\otimes(n) &= 6\mathcal{S}_\otimes(n/3), \\ \mathcal{D}(n) &= \mathcal{D}(n/3) + 3D_\oplus. \end{cases} \qquad \begin{cases} \mathcal{S}_\oplus(n) &= \frac{16}{3}n^{\log_3(6)} - \frac{22n}{3} + 2, \\ \mathcal{S}_\otimes(n) &= n^{\log_3(6)}, \\ \mathcal{D}(n) &= 3\log_3(n)D_\oplus + D_\otimes. \end{cases}$$

### 2.3 Review of four-way split formula with nine recursive multiplications

In [1], Bernstein has shown that it is possible to obtain further improvement in polynomial multiplication when $n = 2^k$ by considering four-way split formula. We review here this approach. Let $A$ and $B$ be two polynomials of degree $n - 1$, which are split in four $A = A_0 + A_1X^{n/4} + A_2X^{2n/4} + A_3X^{3n/4}$ and $B = B_0 + B_1X^{n/4} + B_2X^{2n/4} + B_3X^{3n/4}$, where $A_i$ and $B_i$ have degree $n/4 - 1$ for $i = 0, 1, 2, 3$. The idea of Bernstein is to apply two recursions the two-way split formula and then optimize the reconstruction part of the resulting formula. A first recursion re-expresses the product $C = A \times B$ into three products $(A_0 + A_1X^{n/4})(B_0 + B_1X^{n/4})$, $(A_2 + A_3X^{n/4})(B_2 + B_3X^{n/4})$ and $((A_0 + A_2) + (A_1 + A_3)X^{n/4})((B_0 + B_2) + (B_1 + B_3)X^{n/4})$. If we again apply the two-way split formula to each of these products we obtain nine products of polynomials of size $n/4$ as follows and as shown in Table 2:

- $(A_0 + A_1X^{n/4})(B_0 + B_1X^{n/4})$ yields the products $P_0, P_1$ and $P_2$,
- $(A_2 + A_3X^{n/4})(B_2 + B_3X^{n/4})$ yields the products $P_3, P_4$ and $P_5$,

- $((A_0 + A_2) + (A_1 + A_3)X^{n/4})((B_0 + B_2) + (B_1 + B_3)X^{n/4})$ yields the products $P_6, P_7$ and $P_8$,

We then apply two recursions of the Karatsuba reconstruction which results in the following expression of $C$ in terms of $P_i, i = 0, \ldots, 8$.

$$\begin{aligned} C &= \left(P_0(1 + X^{n/4}) + P_1(X^{n/4} + X^{2n/4}) + P_2X^{n/4}\right)(1 + X^{n/2}) \\ &\quad \left(P_3(1 + X^{n/4}) + P_4(X^{n/4} + X^{2n/4}) + P_5X^{n/4}\right)(X^{n/2} + X^{2n/2}) \\ &\quad \left(P_6(1 + X^{n/4}) + P_7(X^{n/4} + X^{2n/4}) + P_8X^{n/4}\right)X^{n/2} \end{aligned}$$

Bernstein has proposed to arrange the previous expression as follows

$$\begin{aligned} C &= \left(\left(P_0 + P_1X^{n/4} + P_3X^{2n/4} + P_4X^{3n/4}\right)(1 + X^{n/4}) + X^{n/4}P_2 + X^{3n/4}P_5\right)(1 + X^{n/2}) \\ &\quad + \left((P_6 + P_7X^{n/4})(1 + X^{n/4}) + P_8X^{n/4}\right)X^{n/2}. \end{aligned}$$

This latter expression leads to the reconstruction formula given in Table 2.

TABLE 2

Bernstein's four-way split formula with nine recursive multiplications [1]

| Operations | Computations | Cost | | |
|---|---|---|---|---|
| | | $\#\oplus$ | $\#\otimes$ | Delay |
| Rec. Prod. | $P_0 = A_0B_0$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_1 = A_1B_1$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_2 = (A_0 + A_1)(B_0 + B_1)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_3 = A_2B_2$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_4 = (A_2 + A_3)(B_2 + B_3)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_5 = A_3B_3$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_6 = (A_0 + A_2)(B_0 + B_2)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_7 = (A_1 + A_3)(B_1 + B_3)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_8 = (A_0 + A_1 + A_2 + A_3)$ $\times(B_0 + B_1 + B_2 + B_3)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $2D_\oplus + \mathcal{D}(n/4)$ |
| Reconst. | $R_0 = P_0 + P_1X^{n/4}$ $+P_3X^{2n/4} + P_4X^{3n/4}$ | $3n/4 - 3$ | 0 | $D_\oplus$ |
| | $R_1 = (1 + X^{n/4})R_0$ | $4n/4 - 1$ | 0 | $D_\oplus$ |
| | $R_2 = R_1 + X^{n/4}P_2 + X^{3n/4}P_5$ | $4n/4 - 2$ | 0 | $D_\oplus$ |
| | $R_3 = P_6 + X^{n/4}P_7$ | $n/4 - 1$ | 0 | $D_\oplus$ |
| | $R_4 = R_3(1 + X^{n/4})$ | $2n/4 - 1$ | 0 | $D_\oplus$ |
| | $R_5 = R_4 + X^{n/4}P_8$ | $2n/4 - 1$ | 0 | $D_\oplus$ |
| | $R_6 = (1 + X^{2n/4})R_2$ | $4n/4 - 1$ | 0 | $D_\oplus$ |
| | $C = R_6 + X^{2n/4}R_5$ | $4n/4 - 1$ | 0 | $D_\oplus$ |
| Total | | $9\mathcal{S}_\oplus(n/4) + 17n/2 - 11$ | $9\mathcal{S}_\otimes(n/4)$ | |

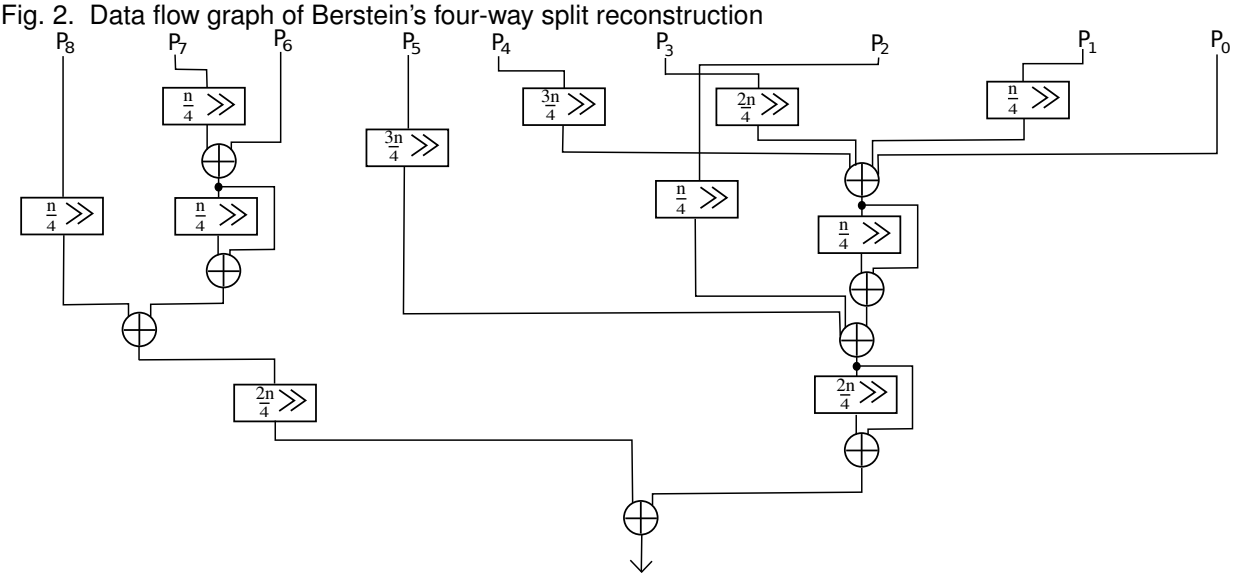
As noticed by Bernstein, the resulting formula requires $3n/2-5$ fewer bit additions than two recursions of the original Karatsuba formulas and $n/4 - 1$ fewer bit additions than the formula of Subsection 2.1.2. The four-way split formula given in Table 2 has the following non-recursive expression when $n$ is an

even power of 2

$$\begin{aligned} S_{\oplus}(n) &= \tfrac{217}{40} n^{\log_2(3)} - \tfrac{34n}{5} - \tfrac{11}{8}, \\ S_{\otimes}(n) &= n^{\log_2(3)}. \end{aligned} \tag{5}$$

The critical path delay is obtained from the data-flow graph of the four-way reconstruction shown in Fig 2. The recursive expression of the delay of Bernstein's four-way multiplier is $\mathcal{D}(n) = 5D_{\oplus} + \mathcal{D}(n/4)$. This results in the non-recursive form of the delay $\mathcal{D}(n) = 2.5\log_2(n)D_{\oplus} + D_{\otimes}$ when $n$ is an even power of 2.

Fig. 2. Data flow graph of Berstein's four-way split reconstruction




## 3 Proposed optimization for three and four-way formulas

In this section we propose some optimizations for three and four-way split formulas. The three-way split optimization removes some redundant computations performed in the formulas presented in 2.2 resulting in a better space complexity for each case. We then present a four-way split formula which combines Bernstein's four-way split approach and the overlap free approach. This formula achieves the delay of the two-way formula of Fan *et al.* with a smaller space complexity.

### 3.1 Improvement on the space complexity of three-way multiplication

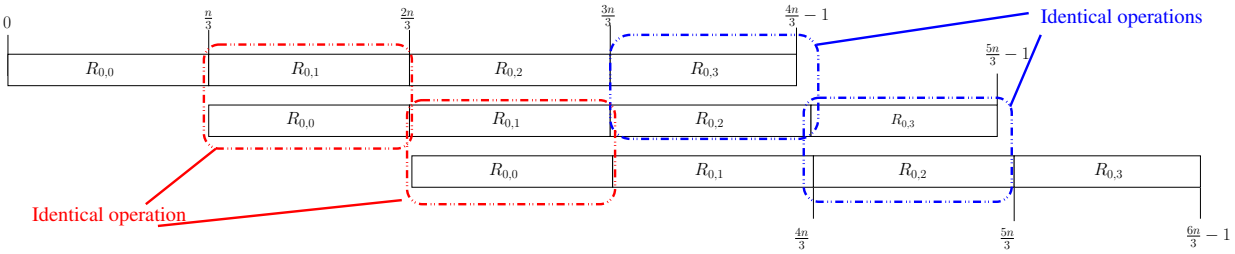
In this subsection, we present an optimized version of the formula presented in Table 1. We keep the six recursive products unchanged and optimize only the reconstruction part of the formula.We recall that the six products in Table 1 results in six terms $P_0, P_1, \ldots, P_5$ of degree $2n/3 - 2$ and the product $C$ is

reconstruct as follows

$$\begin{aligned} R_0 &= (P_0 + X^{n/3}P_1 + X^{2n/3}P_2), \\ R_1 &= R_0(1 + X^{n/3} + X^{2n/3}), \\ C &= R_1 + P_4X^{n/3} + P_5X^{2n/3} + P_3X^{3n/3}. \end{aligned}$$

The computation in $R_1$, which has a cost of $2n-2$ bit additions as shown in Table 1, can be slightly optimized. For this, after spliting we write $R_0 = R_{0,0} + R_{0,1}X^{n/3} + R_{0,2}X^{2n/3} + R_{0,3}X^{3n/3}$ and we note that, based on Fig. 3, there are two sums of two terms which appear twice during the computation of $R_1 = R_0 + X^{n/3}R_0 + X^{2n/3}R_0$.

Fig. 3. Redundant computation in three-way reconstruction




We can modify the reconstruction in order to save each of these re-occuring two term sums. For this, we split the three products $P_0, P_1$ and $P_2$ in two parts $P_0 = P_{0,L} + X^{n/3}P_{0,H}$, $P_1 = P_{1,L} + X^{n/3}P_{1,H}$ and $P_2 = P_{2,L} + X^{n/3}P_{2,H}$ where $P_{i,L}$ and $P_{i,H}$ are degree $n/3-2$ polynomials for $i = 0, 1, 2$. We then express $R_0$ and $R_1$ in terms of $P_{i,L}$ and $P_{i,H}$ for $i = 0, 1, 2$. The last computation for $C$ remains unchanged. The resulting modified reconstruction formula is given in Table 3.

TABLE 3

Space efficient three-way split reconstruction

| Operations | Computations | ($\#\oplus$) |
|---|---|---|
| Computations for $R_0$ | $R_{0,1} = P_{0,H} + P_{1,L}$ | $n/3-1$ |
| | $R_{0,2} = P_{1,H} + P_{2,L}$ | $n/3-1$ |
| Computations for $R_1$ | $R_{1,1} = P_{0,L} + R_{0,1}$ | $n/3$ |
| | $R_{1,2} = R_{0,2} + R_{1,1}$ | $n/3$ |
| | $R_{1,4} = P_{2,H} + R_{0,2}$ | $n/3-1$ |
| | $R_{1,3} = R_{1,4} + R_{0,1}$ | $n/3$ |
| Computations for $C$ | $C = (P_{0,L} + R_{1,1}X^{n/3} + R_{1,2}X^{2n/3} + R_{1,3}X^{3n/3} + R_{1,4}X^{4n/3} + P_{2,H}X^{5n/3})$ $+P_3X^{n/3} + P_4X^{2n/3} + P_5X^{3n/3}$ | $3(2n/3-1)$ |
| Total | | $\frac{12n}{3}-6$ |

If we consider the overall optimized formula which comprises the recursive products of Table 1 plus the proposed optimized reconstruction of Table 3, we obtain the recursive and non-recursive forms of

the complexities given in (6). Note that the modification done on the formula does not affect the delay, so it is unchanged.

$$\begin{cases} \mathcal{S}_{\oplus}(n) &= 6\mathcal{S}_{\oplus}(n/3) + 6n - 6, \\ \mathcal{S}_{\otimes}(n) &= 6\mathcal{S}_{\otimes}(n/3), \\ \mathcal{D}(n) &= \mathcal{D}(n/3) + 4. \end{cases} \qquad \begin{cases} \mathcal{S}_{\oplus}(n) &= \frac{24}{5}n^{\log_3(6)} - 6n + 6/5, \\ \mathcal{S}_{\otimes}(n) &= n^{\log_3(6)}, \\ \mathcal{D}(n) &= 4\log_3(n)D_{\otimes} + D_{\oplus}. \end{cases} \tag{6}$$

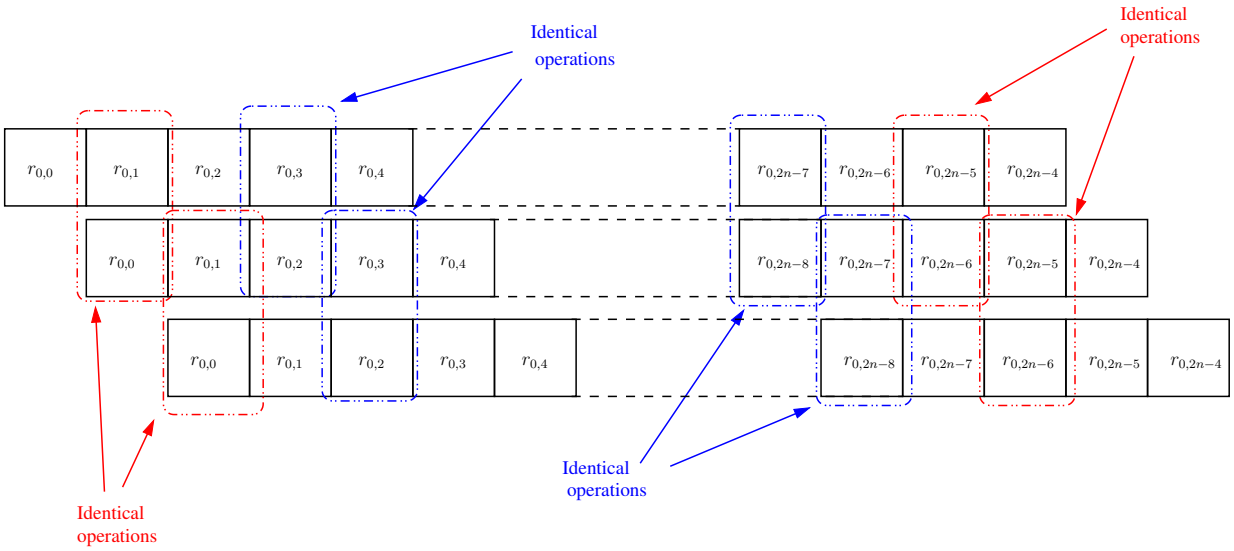
### 3.2 Improvement of the overlap free three-way formula

In this subsection, we present an optimization of the overlap free three-way split formula (cf. Subsection 2.2.2). We do by leaving the recursive multiplications of the formula unchanged. Let $P_0(Y), \ldots, P_5(Y)$ be the six products. We solely modify the sequence of operations in the reconstruction: we propose to perform the reconstruction of $C$ as follows

$$C = (P_0(X^3) + XP_1(X^3) + X^2P_2(X^3))(1 + X + X^2) + XP_3(^3) + X^2P_4(X^3) + X^3P_5(X^3)$$

We then denote $R_0 = P_0(X^3) + XP_1(X^3) + X^2P_2(X^3)$ and $R_1 = (P_0(X^3) + XP_1(X^3) + X^2P_2(X^3))(1 + X + X^2) = R_0 \times (1 + X + X^2)$. We note that the computation of $R_0$ consists of interleaving the coefficients of $P_0(X^3), P_1(X^3)$ and $P_2(X^3)$. This does not require any logic gate. Then we write $R_0 = \sum_{i=0}^{2n-4} r_{0,i}X^i$ and we note that for the computation of $R_1$ we can save some bit operations. Indeed, it is shown in the following figure that some bit additions appear twice in the computation of $R_1$.

Fig. 4. Redundant computation in overlap free three-way reconstruction




Based on this fact, in the computation of $R_1$, we perform only one of these redundant bit additions.

The resulting optimized bit operations for $R_1$ are given below:

$$
\begin{aligned}
&r_{1,0} = r_{0,0}\\
&r_{1,1} = r_{0,0} + r_{0,1}\\
&r_{1,2} = r_{1,1} + r_{0,2}\\
&\textbf{for } i = 1 \textbf{ to } n-3\\
&\quad t_i = r_{0,2i} + r_{0,2i+1}\\
&\quad r_{1,2i+1} = t_i + r_{0,2i-1}\\
&\quad r_{1,2i+2} = t_i + r_{0,2i+2}\\
&\textbf{end for}\\
&r_{1,2n-3} = r_{2n-5} + r_{2n-4}\\
&r_{1,2n-2} = r_{2n-4}
\end{aligned}
\tag{7}
$$

The final operation $C = R_1 + XP_3(X^3) + X^2P_4(X^3) + X^3P_5(X^3)$ are done in a straightforward way: the operation $XP_3(X^3)+X^2P_4(X^3)+X^3P_5(X^3)$ consists of interleaving the coefficients of $P_3, P_4$ and $P_5$ and then we just have to add the resulting polynomial to $R_1$.

*Space complexity.* The recursive products have the same cost as given in Subsection 2.2.2, i.e., $2n+6\mathcal{S}_\oplus(n)$ bit additions and $6\mathcal{S}_\oplus(n)$ bit multiplications. The proposed reconstruction formula requires no bit operations for $R_0$, but $(2+3(n-3)+1)$ bit additions for the computations of $R_1$ and $3(2n/3-1)$ for the final addition in $C$. Consequently, the proposed three-way split multiplier has the following space complexity

$$
\begin{cases}
\mathcal{S}_\oplus(n) &= 6\mathcal{S}_\oplus(n/3) + 7n - 9,\\
\mathcal{S}_\otimes(n) &= 6\mathcal{S}_\otimes(n/3).
\end{cases}
\qquad
\begin{cases}
\mathcal{S}_\oplus(n) &= \frac{26}{5}n^{\log_3(6)} - 7n + 9/5,\\
\mathcal{S}_\otimes(n) &= n^{\log_3(6)}.
\end{cases}
$$

*Delay evaluation.* We evaluate separately the delay for the computation of $R_1$ and the delay for computation of $XP_3(X^3) + X^2P_4(X^3) + X^3P_5(X^3)$

- We first remark that the delay to compute $R_1$ is equal to the delay of the computation of $P_i, i = 0, 1, 2$ plus $2D_\oplus$ which results in $2D_\oplus + \mathcal{D}(n/3)$.
- The delay required to compute $XP_3(X^3) + X^2P_4(X^3) + X^3P_5(X^3)$ is equal to the delay to compute $P_i$, $i = 3, 4, 5$ and this is equal to $D_\oplus + \mathcal{D}(n/3)$.

Eventually, the critical path delay for the computation of $C$ is equal to $\mathcal{D}(n) = \mathcal{D}(n/3) + 3D_\otimes$ which can be rewritten in the following non-recursive form:

$$\mathcal{D}(n) = 3\log_3(n)D_\oplus + D_\otimes.$$

### 3.3 Proposed four-way split formula

We now propose a four-way split formula which improves the delay of Bernstein's four-way formula, while having a slightly larger space complexity. The proposed formula is obtained by combining the

four way formula of Bernstein and the overlap free method of [3]. Indeed, we use the following special splitting for the two polynomials $A$ and $B$ of degree $n$ :

$$\begin{aligned} A &= A_0(X^4) + XA_1(X^4) + X^2A_2(X^4) + X^3A_3(X^4), \\ B &= B_0(X^4) + XB_1(X^4) + X^2B_2(X^4) + X^3B_3(X^4), \end{aligned}$$

where $A_i(Y) = \sum_{j=0}^{n/4-1} a_{i+4j}Y^j$ and $B_i(Y) = \sum_{j=0}^{n/4-1} b_{i+4j}Y^j$ are each a polynomial of degree $n/4-1$ in $Y$. Then we can consider $A = A_0(Y) + XA_1(Y) + X^2A_2(Y) + X^3A_3(Y)$ and $B = B_0(Y) + XB_1(Y) + X^2B_2(Y) + X^3B_3(Y)$ as polynomials of degree 3 in $X$. We can then apply the four-way split formula of Bernstein (Table 2) to compute the product. To obtain the expression of $C(X) = A(X) \times B(X)$ we need to replace $Y$ with $X^4$, this is done just before proceeding to the reconstruction. The resulting modified four-way split formula is given in Table 4.

TABLE 4

Proposed four-way split formulas with nine recursive multiplications

| Operations | Computations | Cost | | |
|---|---|---|---|---|
| | | $\#\oplus$ | $\#\otimes$ | Delay |
| Rec. Prod. | $P_0(Y) = A_0B_0$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_1(Y) = A_1B_1$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_2(Y) = (A_0 + A_1)(B_0 + B_1)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_3(Y) = A_2B_2$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_4(Y) = A_3B_3$ | $S_\oplus(n/4)$ | $S_\otimes(n/4)$ | $\mathcal{D}(n/4)$ |
| | $P_5(Y) = (A_2 + A_3)(B_2 + B_3)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_6(Y) = (A_0 + A_2)(B_0 + B_2)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_7(Y) = (A_1 + A_3)(B_1 + B_3)$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $D_\oplus + \mathcal{D}(n/4)$ |
| | $P_8(Y) = ((A_0 + A_1) + (A_2 + A_3))((B_0 + B_1) + (B_2 + B_3))$ | $S_\oplus(n/4) + n/2$ | $S_\otimes(n/4)$ | $2D_\oplus + \mathcal{D}(n/4)$ |
| Reconst. | $R_0 = P_0(X^4) + P_1(X^4)X + P_3(X^4)X^2 + P_4(X^4)X^3$ | 0 | 0 | 0 |
| | $R_1 = (1 + X)R_0$ | $4n/2 - 5$ | 0 | $D_\oplus$ |
| | $R_2 = R_1 + XP_2(X^4) + X^3P_5(X^4)$ | $2n/2 - 2$ | 0 | $D_\oplus$ |
| | $R_3 = P_6(X^4) + XP_7(X^4)$ | 0 | 0 | 0 |
| | $R_4 = R_3(1 + X)$ | $n/2 - 1$ | 0 | $D_\oplus$ |
| | $R_5 = R_4 + XP_8(X^4)$ | $n/2 - 1$ | 0 | $D_\oplus$ |
| | $R_6 = (1 + X^2)R_2$ | $4n/2 - 5$ | 0 | $D_\oplus$ |
| | $C = R_6 + X^2R_5$ | $3n/2 - 3$ | 0 | $D_\oplus$ |
| | | $9\mathcal{S}_\oplus(n/4) + 20n/2 - 17$ | $9\mathcal{S}_\otimes(n/4)$ | |

*Space complexity.* The costs of the recursive products are unchanged compared to Table 2: only the costs of the steps of the reconstruction, which are different, are to be determined. In Table 4 we detail these costs and derive the recursive form of the overall complexity. The resulting non-recursive form of the

complexity is given below:

$$\begin{cases} S_{\oplus}(n) &= \frac{47}{8} n^{\log_2(3)} - 8n + 17/8, \\ S_{\otimes}(n) &= n^{\log_2(3)}. \end{cases} \tag{8}$$

*Time complexity.* We now evaluate the delay complexity of the proposed four-way formula. The data-flow graph of the reconstruction is the same as that of the four-way formula of Table 2 shown in Fig. 2. The only difference are in the various shift operations in the data-flow graph and in the cost of the corresponding steps. We deduce that the critical path delay is $\mathcal{D}(n) = 4D_{\oplus} + \mathcal{D}(n/4)$ which results in $\mathcal{D}(n) = 4\log_4(n)D_{\oplus} + D_{\otimes} = 2\log_2(n)D_{\oplus} + D_{\otimes}$.

## 4 Block recombination for polynomial multiplication

In this section we present a block recombination approach which reduces the space complexity of the multipliers presented in Section 2 and 3. This approach is an extension of the approach presented in [5] for Toeplitz matrix-vector multiplication.

### 4.1 Decomposition of polynomial multiplication algorithms

A polynomial multiplication algorithm can be divided into three separate computations: the component polynomial formation (CPF), the component multiplication (CM) and the reconstruction (R). Here, we explain these computations for the two and three-way split formulas described in Sections 2 and 3. The four-way split approach can be seen as an optimized version of the two-way formula. Consequently, we will not differentiate two and four-way split multipliers and, later in this paper, complexity results of these multipliers are presented in the same table.

- *Component polynomial formation (CPF).* Let $A$ be a polynomial of size $n = 2^k$ in the two-way case and $n = 3^k$ in the three-way case. We recursively define the component polynomial formation as follows

$$CPF_{2,n}(A) = \begin{cases} A \text{ if } n = 1, \\ [CPF_{2,n/2}(A_0), CPF_{2,n/2}(A_0 + A_1), CPF_{2,n/2}(A_1)] \text{ if } n = 2^k > 1 \text{ and } A = A_0 + X^{n/2}A_1. \end{cases}$$

$$CPF_{3,n}(A) = \begin{cases} A \text{ if } n = 1, \\ [CPF_{3,n/3}(A_0), CPF_{3,n/3}(A_1), CPF_{3,n/3}(A_2), CPF_{3,n/3}(A_1 + A_2), CPF_{3,n/3}(A_0 + A_1), \\ \quad CPF_{3,n/3}(A_0 + A_2)] \text{ if } n = 3^k > 1 \text{ and } A = A_0 + X^{n/3}A_1 + X^{2n/3}A_2. \end{cases} \tag{9}$$

  We can similarly define $CPF_{2,n}$ and $CPF_{3,n}$ functions for overlap free formulas: this requires us to replace the usual way of splitting by the odd-even splitting used in overlap free formulas.
  In the sequel, we will often refer to the *component representation* of a degree $n$ polynomial $A$ as the bit array $\hat{A} = CPF_{2,n}(A)$ (resp. $\hat{A} = CPF_{3,n}(A)$) which has a bit length of $n^{\log_2(3)}$ (resp. $n^{\log_3(6)}$).
- *Component multiplication (CM).* The component multiplication consists of the bit-wise multiplication of the component polynomial formations $\hat{A}$ and $\hat{B}$ of two polynomials $A$ and $B$

$$CM(A, B) = \hat{A} \otimes \hat{B}$$

This operation can be implemented with $n^{\log_2(3)}$ parallel AND gates in the case of two-way split approaches and $n^{\log_3(6)}$ parallel AND gates in the case of three-way split approaches.

- *Reconstruction (R).* The reconstruction $R(\hat{C})$ of a vector $\hat{C}$ of length $n^{\log_2(3)}$ (resp. $n^{\log_3(6)}$) consists of recursively applying the reconstruction part of the considered two or four-way (resp. three way) split formula of Subsections 2.1 , 2.3 (resp. Subsections 2.2.1 or 3.1) to obtain a polynomial $C$ of degree $2n-2$. For example, the function $R$ of the original Karatsuba formula of Subsection 2.1.1 and the three-way split formula of Subsection 2.2.1 are as follows

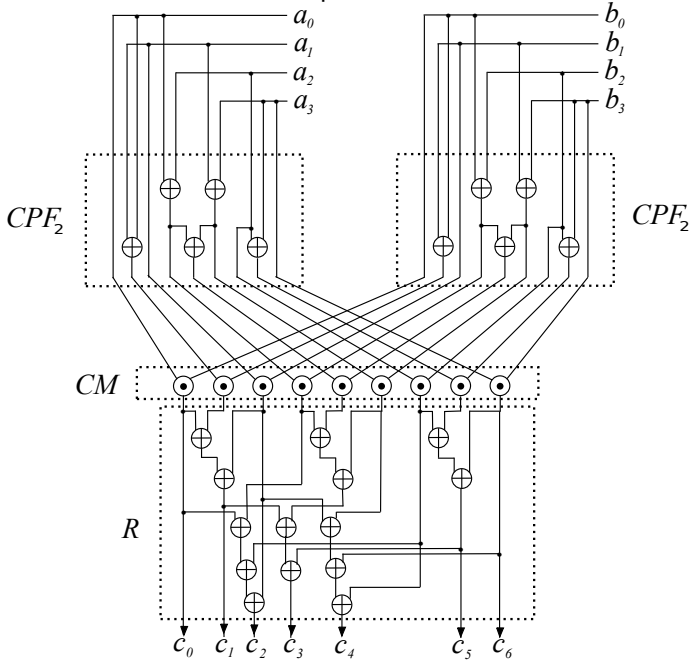
$$R_{2,n}(\hat{C}) = \begin{cases} \hat{C} \text{ if } n = 1, \\ R_{2,n/2}(\hat{C}_0) + (R_{2,n/2}(\hat{C}_1) + R_{2,n/2}(\hat{C}_0) + R_{2,n/2}(\hat{C}_2))X^{n/2} + R_{2,n/2}(\hat{C}_2)X^n \\ \qquad \text{if } |\hat{C}| > 1 \text{ and } \hat{C} = [\hat{C}_0, \hat{C}_1, \hat{C}_2] \text{ with } |\hat{C}_i| = |\hat{C}|/3. \end{cases} \tag{10}$$

$$R_{3,n}(\hat{C}) = \begin{cases} \hat{C} \text{ if } n = 1, \\ (R_{3,n/3}(\hat{C}_0) + R_{3,n/3}(\hat{C}_1)X^{n/3} + R_{3,n/3}(\hat{C}_2)X^{2n/3})(1 + X^{n/3} + X^{2n/3}) + R_{3,n/3}(\hat{C}_3)X^{n/3} \\ \quad + R_{3,n/3}(\hat{C}_4)X^{2n/3} + R_{3,n/3}(\hat{C}_5)X^{3n/3} \text{ if } |\hat{C}| > 1 \text{ and } \hat{C} = [\hat{C}_0, \hat{C}_1, \hat{C}_2, \hat{C}_3, \hat{C}_4, \hat{C}_5] \\ \quad \text{with } |\hat{C}_i| = |\hat{C}|/6. \end{cases} \tag{11}$$

Similar functions $R_{2,n}$ and $R_{3,n}$ can be defined for each formula given in Sections 2 and 3.

We illustrate, in Fig. 5, the block decomposition of a parallel multiplier for $n = 4$ based on the original Karatsuba formula reviewed in Subsection 2.1.1.

Fig. 5. Block decomposition of the Karatsuba multiplication circuit for $n = 4$




In the sequel, we will recombine the blocks of the multiplier in order to reduce the space complexity. But first, we establish the complexity of each block of the multiplier for various formulas presented in

Sections 2 and 3.

**Lemma 1** (Block complexities)**.** *We consider a multiplier based on one of the formulas of Section 2 or 3. Let $\mathcal{S}_{\oplus}(n)$ be the number of bit additions involved in the multiplication, then the block complexities are as follows:*

*(i) For the two or the four-way split formulas we have*

$$\begin{aligned}\mathcal{S}^{CPF}_{2,\oplus}(n) &= n^{\log_2(3)} - n\\ \mathcal{S}^{CM}_{2,\otimes}(n) &= n^{\log_2(3)}\\ \mathcal{S}^{R}_{2,\oplus}(n) &= \mathcal{S}_{\oplus}(n) - 2(n^{\log_2(3)} - n)\end{aligned}$$

*(ii) For the three-way split formulas we have*

$$\begin{aligned}\mathcal{S}^{CPF}_{3,\oplus}(n) &= n^{\log_3(6)} - n\\ \mathcal{S}^{CM}_{3,\otimes}(n) &= n^{\log_3(6)}\\ \mathcal{S}^{R}_{3,\oplus}(n) &= \mathcal{S}_{\oplus}(n) - 2(n^{\log_3(6)} - n)\end{aligned}$$

*Proof:*

- *Complexity of the $CPF$ function.* We first evaluate the complexity of the $CPF$ function in the two-way split case. We assume that $n = 2^k$ and we prove it by induction on $k$. For $k = 0$, i.e., $n = 1$, we have $n^{\log_2(s)} - n = 1 - 1 = 0$ which is correct and the formula of the lemma holds. Now, we assume that the expression of $\mathcal{S}^{CPF}_{2,\oplus}$ is true for $k$ and we show it is also true for $k + 1$. By definition of $CPF_{2,n}$ we have

$$CPF_{2,n}(A) = [CPF_{2,n/2}(A_0), CPF_{2,n/2}(A_0 + A_1), CPF_{2,n/2}(A_1)]$$

  where $A_0$ and $A_1$ are the polynomials resulting from the splitting (a regular two-way splitting or an even/odd splitting when overlap free method is used). In other words, the computation of $CPF_{2,n}(A)$ requires $n/2$ bit additions for $A_0 + A_1$ plus $3\mathcal{S}^{CPF}_{2,\oplus}(n/2)$. By induction hypothesis, we have $\mathcal{S}_{CPF,\oplus}(n/2) = \left(\frac{n}{2}\right)^{\log_2(3)} - \frac{n}{2}$. This results in

$$\mathcal{S}^{CPF}_{2,\oplus}(n) = n/2 + 3(\left(\frac{n}{2}\right)^{\log_2(3)} - \frac{n}{2}) = n^{\log_2(3)} - n,$$

  This ends the proof for the two-way split case. The proof for the three-way split case is similar, for the sake of brevity, we skip the details.
- *Complexity of the $CM$ function.* We note that CM function consists of all the bit multiplications of the multiplier. This means that its complexity is equal to number of bit multiplications $\mathcal{S}_{\otimes}(n)$. The value of $\mathcal{S}^{CM}_{2,\otimes}(n)$ (resp. $\mathcal{S}^{CM}_{3,\otimes}(n)$) is a direct consequence of this remark.
- *Complexity of the $R$ function.* We have the following relation between the complexity $\mathcal{S}_{s,\oplus}(n), s \in \{2, 3\}$, of the multiplier and the complexity of the block $CPF$ and $R$

$$\mathcal{S}_{s,\oplus}(n) = 2\mathcal{S}^{CPF}_{s,\oplus}(n) + \mathcal{S}^{R}_{s,\oplus}(n) \text{ for } s \in \{2, 3\}.$$

  Consequently we have $\mathcal{S}^{R}_{s,\oplus}(n) = \mathcal{S}_{s,\oplus}(n) - 2\mathcal{S}^{CPF}_{s,\oplus}(n)$, and the value stated in the lemma is obtained after replacing the value of $\mathcal{S}^{CPF}_{s,\oplus}(n)$ with $n^{\delta} - n$ where $\delta = \log_2(3)$ or $\delta = \log_3(6)$ given in $(i)$.

□

Using the previous lemma, we compute the complexity of the block $R$ for the formulas presented in Sections 2 and 3. The results are reported in Table 5.

TABLE 5
Space complexity of the reconstruction blocks

| Method | Split | Complexity |
|---|---|---|
| Subsection 2.1.1 | 2 | $4n^{\log_2(3)} - 6n + 2$ |
| Subsection 2.1.2 | 2 | $\frac{7}{2}n^{\log_2(3)} - 5n + \frac{3}{2}$ |
| Subsection 2.1.3 | 2 | $4n^{\log_2(3)} - 6n + 2$ |
| Subsection 2.2.1 | 3 | $\frac{49}{15}n^{\log_3(6)} - \frac{14n}{3} + \frac{7}{5}$ |
| Subsection 2.2.2 | 3 | $\frac{10}{3}n^{\log_3(6)} - \frac{16n}{3} + 2$ |
| Subsection 3.1 | 3 | $\frac{14}{5}n^{\log_3(6)} - 4n + \frac{6}{5}$ |
| Subsection 3.2 | 3 | $\frac{16}{5}n^{\log_3(6)} - 5n + \frac{9}{5}$ |
| Subsection 2.3 | 4 | $\frac{137}{40}n^{\log_2(3)} - \frac{24}{5}n - \frac{11}{8}$ |
| Subsection 3.3 | 4 | $\frac{31}{8}n^{\log_2(3)} - 6n - \frac{17}{8}$ |

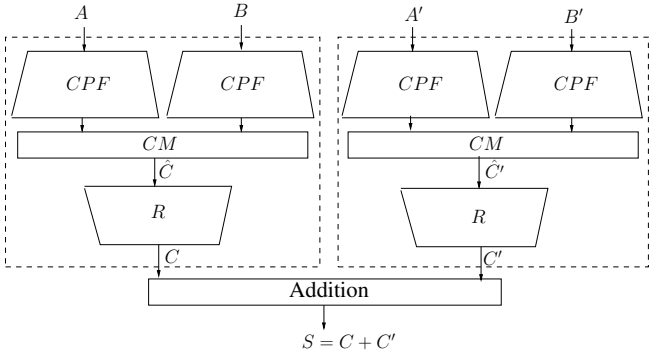
### 4.2 Block recombination of two multiplications and an add

In this subsection, we state the basic block recombination operation, which we will use later to recombine the two and three-way multipliers. Specifically, we consider the problem of performing two instances of polynomial multiplications in parallel followed by an addition. In other words, if $A, A', B$ and $B'$ are polynomials of degree $n-1$ each, we want to compute

$$S = A \times B + A' \times B'.$$

A straightforward approach to design a parallel architecture which computes $S$ is shown in Fig. 6. In

Fig. 6. Two-multiplications-and-an-add architecture



order to reduce the space complexity of the double multiplication and an addition shown in Fig. 6, we recombine the block of Fig 6. This recombination is based on the following lemma which presents results for both two- and three-way split cases, with the latter appearing in brackets.

**Lemma 2.** *Let $\hat{C}$ and $\hat{C}'$ be two arrays of $n^{\log_2(3)}$ bits (resp. $n^{\log_3(6)}$ bits). Let $R_{2,n}$ (resp. $R_{3,n}$) be the reconstruction function, then we have*

$$R_{2,n}(\hat{C}) + R_{2,n}(\hat{C}') = R_{2,n}(\hat{C} + \hat{C}') \qquad (\textit{resp. } R_{3,n}(\hat{C}) + R_{3,n}(\hat{C}') = R_{3,n}(\hat{C} + \hat{C}')). \tag{12}$$

*Proof:* We prove this lemma only for the reconstruction function of the original (two-way) Karatsuba formula (Subsection 2.1.1). We give the proof by induction on $k$ where $n = 2^k$. For $k = 0$ we have $n = 1$ and in this case, by definition of $R_{2,n}$, we have $R_{2,n}(\hat{C}) = \hat{C},\ R_{2,n}(\hat{C}') = \hat{C}'$ and $R_{2,n}(\hat{C} + \hat{C}') = \hat{C} + \hat{C}'$. This means that (12) is satisfied for $k = 0$. Now we assume that Lemma 2 is true for $k$ and we prove it for $k + 1$. We split $\hat{C}$ and $\hat{C}'$ in three parts of length $\frac{n^{\log_2(3)}}{3}$

$$\hat{C} = [\hat{C}_0, \hat{C}_1, \hat{C}_2] \text{ and } \hat{C}' = [\hat{C}'_0, \hat{C}'_1, \hat{C}'_2].$$

Then, we have $\hat{C} + \hat{C}' = [\hat{C}_0 + \hat{C}'_0, \hat{C}_1 + \hat{C}'_1, \hat{C}_2 + \hat{C}'_2]$ and by definition of $R_{2,n}$ we have

$$\begin{array}{rcl} R_{2,n}(\hat{C}) & = & R_{2,n/2}(\hat{C}_0) + (R_{2,n/2}(\hat{C}_1) + R_{2,n/2}(\hat{C}_0) + R_{2,n/2}(\hat{C}_2))X^{n/2} + R_{2,n/2}(\hat{C}_2)X^n, \\ R_{2,n}(\hat{C}') & = & R_{2,n/2}(\hat{C}'_0) + (R_{2,n/2}(\hat{C}'_1) + R_{2,n/2}(\hat{C}'_0) + R_{2,n/2}(\hat{C}'_2))X^{n/2} + R_{2,n/2}(\hat{C}'_2)X^n, \\ R_{2,n}(\hat{C} + \hat{C}') & = & R_{2,n/2}(\hat{C}_0 + \hat{C}'_0) + (R_{2,n/2}(\hat{C}_1 + \hat{C}'_1) + R_{2,n/2}(\hat{C}_0 + \hat{C}'_0) + R_{2,n/2}(\hat{C}_2 + \hat{C}'_2))X^{n/2} \\ & & + R_{2,n/2}(\hat{C}_2 + \hat{C}'_2)X^n. \end{array} \tag{13}$$

We then add these expressions of $R_{2,n}(\hat{C})$ and $R_{2,n}(\hat{C}')$ and apply the induction hypothesis for each terms $R_{2,n}(\hat{C}_i) + R_{2,n}(\hat{C}'_i), i = 0, 1, 2$. We obtain

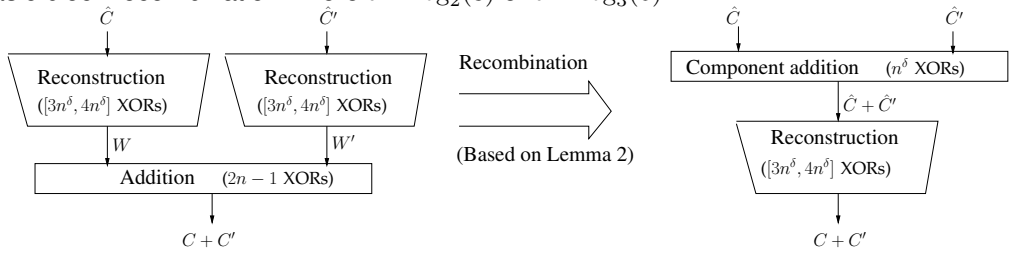
$$\begin{array}{rcl} R_{2,n}(\hat{C}) + R_{2,n}(\hat{C}') & = & (R_{2,n/2}(\hat{C}_0) + R_{2,n/2}(\hat{C}'_0)) + \Big(R_{2,n/2}(\hat{C}_1) + R_{2,n/2}(\hat{C}'_1) + R_{2,n/2}(\hat{C}_0) + R_{2,n/2}(\hat{C}'_0) \\ & & + R_{2,n/2}(\hat{C}_2) + R_{2,n/2}(\hat{C}'_2)\Big) X^{n/2} + (R_{2,n/2}(\hat{C}_2) + R_{2,n/2}(\hat{C}'_2))X^n \\ & = & R_{2,n/2}(\hat{C}_0 + \hat{C}'_0) + (R_{2,n/2}(\hat{C}_1 + \hat{C}'_1) + R_{2,n/2}(\hat{C}_0 + \hat{C}'_0) + R_{2,n/2}(\hat{C}_2 + \hat{C}'_2))X^{n/2} \\ & & + R_{2,n/2}(\hat{C}_2 + \hat{C}'_2)X^n. \end{array}$$

From (13), we see that this latest expression is equal to $R(\hat{C} + \hat{C}')$, and this ends the proof for the case of the Karatsuba reconstruction. The proof for the other cases, i.e., other two-, three-, four-way split reconstructions can be obtained in a similar way. For the sake of simplicity we skip here details of these proofs. □

The previous lemma shows that the sequence of operations *reconstruction followed by an addition* can be reversed, i.e., we can first perform an addition of the original input components and then perform the reconstruction on the sum. In Fig. 7, we apply this property to recombine the lower two layers of blocks of the two-multiplication-and-an-add architecture (Fig. 6).

In Fig. 7 we also provide the sizes of different blocks: an addition of input components (CA) requires $n^\delta$ XOR gates, where $\delta$ is equal to $\log_2(3)$ for the two-way and is $\log_3(6)$ for the three-way split multipliers.

Fig. 7. Basic block recombination where $\delta = \log_2(3)$ or $\delta = \log_3(6)$




Similarly, we note that the reconstruction block has a space complexity in the interval $[3n^\delta, 4n^\delta]$ (see Table 5). The recombination replaces a block consisting of at least $3n^\delta$ XOR gates by a block that has only $n^\delta$ XOR gates. This means that we save at least $2n^\delta$ XOR gates.

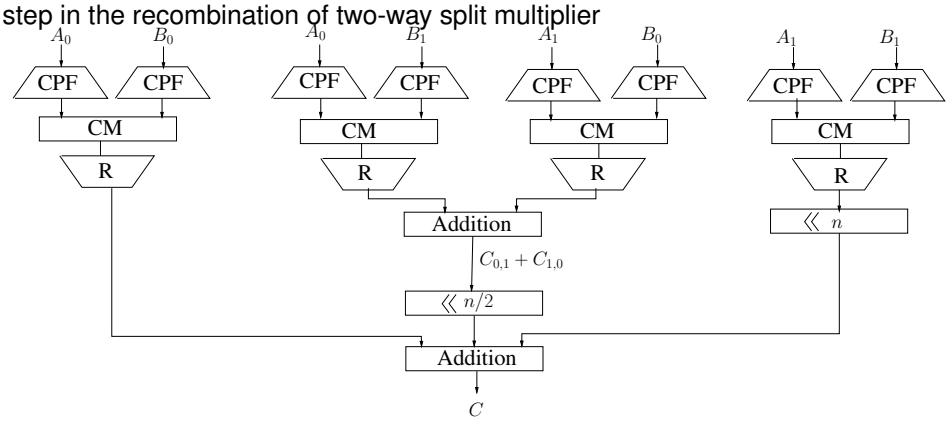
### 4.3 Two-way split block recombination

In this subsection we present a block recombination approach for the two-way split multiplier in order to reduce the space complexity. Let $A$ and $B$ be two degree $n-1$ polynomials where $n > 1$ is power of 2. We split $A, B$ in two parts $A = A_0 + A_1 X^{n/2}$ and $B = B_0 + B_1 X^{n/2}$, where $A_i$ and $B_i$ have degree $n/2 - 1$ for $i = 0, 1$. We then expand the product $A \times B$ as follows

$$AB = A_0 B_0 + (A_0 B_1 + A_1 B_0) X^{n/2} + A_1 B_1 X^n. \tag{14}$$

We can design an architecture based on the previous expression of $C = AB$ which independently performs each product $C_{i,j} = A_i B_j$ through a two-way split multiplier. After that we reconstruct $C$ by performing the additions and shifts corresponding to the expression $C = C_{0,0} + (C_{0,1} + C_{1,0}) X^{n/2} + C_{1,1} X^n$. The resulting architecture is depicted in Fig. 8.

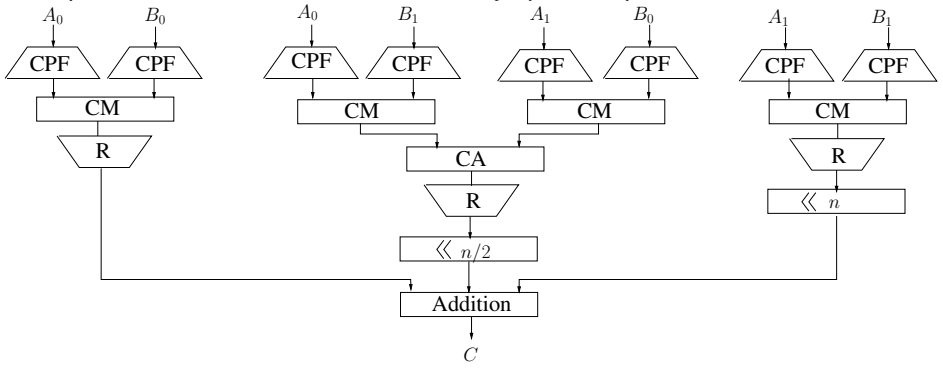
Fig. 8. First step in the recombination of two-way split multiplier




We then note that the computation of $C_{0,1} + C_{1,0}$ in Fig. 8 consists of two parallel multiplications

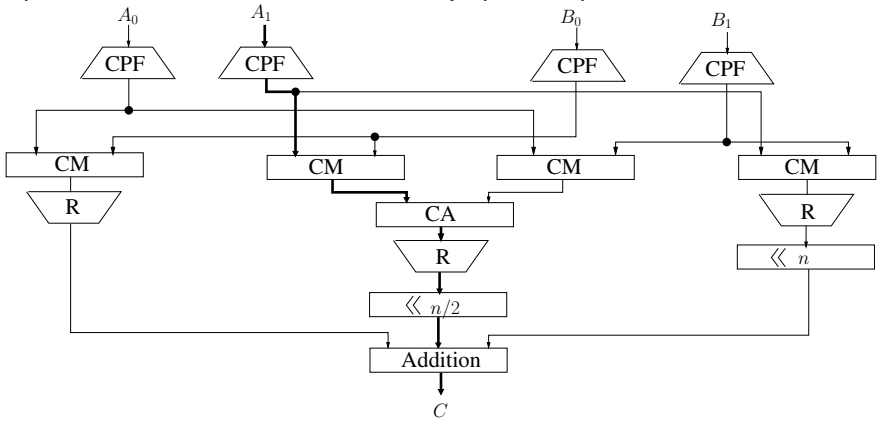
followed by an addition. We can apply the block recombination presented in Subsection 4.2 which reverses the order of the reconstruction and the addition. The resulting architecture is shown in Fig. 9.

Fig. 9. Second step in the recombination of the two-way split multiplier




We finally perform a last modification in the previous recombined architecture: we remove the redundant blocks $CVF_{2,n/2}(A_i)$, $i = 0, 1$ and $CVF_{2,n/2}(B_j)$, $j = 0, 1$ by keeping only one $CVF_{2,n/2}$ block for each $A_i$ and $B_j$ for $i = 0, 1$ and $j = 0, 1$. This results in the architecture shown in Fig. 10.

Fig. 10. Third step in the recombination of the two-way split multiplier




**Complexity of the recombined architecture (Fig 10).** We first evaluate the space complexity of the recombined architecture. We note that the final addition block requires $n-2$ XOR gates since the degree of the polynomials $C_{0,1}$ and $C_{1,0}$ is at most $2n/2-2$. We then express the space complexity in terms of the complexity of each block:

$$\begin{cases} \mathcal{S}_{\oplus}(n) &= 4\mathcal{S}_{\oplus}^{CPF}(n/2) + 3\mathcal{S}_{\oplus}^{R}(n/2) + \mathcal{S}_{\oplus}^{CA}(n/2) + (n-1), \\ \mathcal{S}_{\otimes}(n) &= 4\mathcal{S}_{\otimes}(n/2). \end{cases} \tag{15}$$

In (15), we can replace the terms $\mathcal{S}_{\oplus}^{CPF}(n/2), \mathcal{S}_{\oplus}^{R}(n/2), \mathcal{S}_{\oplus}^{CA}(n/2)$ and $\mathcal{S}_{\otimes}(n/2)$ by their explicit expressions in terms of $n$ given in Lemma 1 and Table 6. This leads to complexity results reported in the middle two columns of Table 6.

TABLE 6

Complexity of two-way recombined multipliers

| Recombined formula | # AND | # XOR | Delay |
|---|---|---|---|
| Original Karatsuba (Subsection 2.1.1) | $\frac{4n^{\log_2(3)}}{3}$ | $\frac{17n^{\log_2(3)}}{3} - 10 + 4$ | $(3\log_2(n) - 1)D_{\oplus} + D_{\otimes}$ |
| Two-way of [1], [10] | $\frac{4n^{\log_2(3)}}{3}$ | $\frac{31n^{\log_2(3)}}{6} - \frac{17n}{2} + \frac{5}{2}$ | $(3\log_2(n) - 1)D_{\oplus} + D_{\otimes}$ |
| Two-way of [3] | $\frac{4n^{\log_2(3)}}{3}$ | $\frac{17n^{\log_2(3)}}{3} - 10 + 4$ | $2\log_2(n)D_{\oplus} + D_{\otimes}$ |
| Four-way of Subsection 2.3 | $\frac{4n^{\log_2(3)}}{3}$ | $\frac{611n^{\log_2(3)}}{120} - \frac{41n}{5} + \frac{17}{8}$ | $(\frac{5}{2}\log_2(n) - \frac{1}{2})D_{\oplus} + D_{\otimes}$ |
| Four-way of Subsection 3.3 | $\frac{4n^{\log_2(3)}}{3}$ | $\frac{133n^{\log_2(3)}}{24} - 10n + \frac{35}{8}$ | $2\log_2(n)D_{\oplus} + D_{\otimes}$ |

We now evaluate the delay of the two-way split multiplier obtained through block recombination. The critical path is shown in bold in Fig. 10. Its delay consists of $2D_{\oplus}$ for the additions (the CA block and the Addition block), plus the delay of the blocks $CPF$, $CA$ and $R$

$$\mathcal{D}(n) = \mathcal{D}_{CPF}(n/2) + \mathcal{D}_{CM}(n/2) + \mathcal{D}_{R}(n/2) + 2D_{\oplus}.$$

We now remark that $\mathcal{D}_{CPF}(n/2)+\mathcal{D}_{CM}(n/2)+\mathcal{D}_{R}(n/2)$ is equal to the delay $\mathcal{D}(n/2)$ of a regular two-way split multiplier of size $n/2$. This results in the complexities reported in the right most column of Table 6 corresponding to the formulas presented in Sections 2 and 3.

### 4.4 Four-way block recombination

In the previous section, we have presented a two-way block recombination. This approach can be generalized to a $2^{\ell}$-way block recombination: the two polynomials are split in $2^{\ell}$ parts, then the product is expanded, and then the products of polynomials of degree $n/2^{\ell}$ are performed in parallel through multiplier of size $n/2^{\ell}$ which are finally recombined. We have evaluated the resulting complexities for various values of $\ell$, and have noticed that the least total gate count (AND and XOR combined) is obtained when $\ell = 2$. We thus only present this optimal case here.

Let $A$ and $B$ be two polynomials of degree $n-1$, where $n = 2^k$ for $k \geq 2$. We split $A$ and $B$ in four $A = A_0 + A_1X^{n/4} + A_2X^{2n/4} + A_3X^{3n/4}$ and $B = B_0 + B_1X^{n/4} + B_2X^{2n/4} + B_3X^{3n/4}$, where $A_i$ and $B_i$ have degree $n/4 - 1$ for $i = 0, 1, 2, 3$. We expand the product as follows:

$$\begin{aligned} AB = &\underbrace{A_0B_0}_{C_0} + \underbrace{(A_0B_1 + A_1B_0)}_{C_1}X^{n/4} + \underbrace{(A_0B_2 + A_1B_1 + A_2B_0)}_{C_2}X^{2n/4} + \underbrace{(A_0B_3 + A_1B_2 + A_2B_1 + A_3B_0)}_{C_3}X^{3n/4} \\ &+ \underbrace{(A_1B_3 + A_2B_2 + A_3B_1)}_{C_4}X^{4n/4} + \underbrace{(A_2B_3 + A_3B_2)}_{C_5}X^{5n/4} + \underbrace{A_3B_3}_{C_6}X^{6n/4}. \end{aligned} \tag{16}$$

Based on the previous equation, we design a multiplier as follows:

1) *Step 1.* we apply the $CPF_{2,n/4}$ function on the polynomials $A_i$ and $B_i$, $i = 0, 1, 2, 3$.
2) *Step 2.* we perform the additions in the expression of $C_1$, $C_2$, $C_3$, $C_4$ and $C_5$ of (16).
3) *Step 3.* we apply the function $R_{2,n/4}$ to the resulting seven terms $\hat{C}_i, i = 0, 1 \ldots, 6$.
4) *Step 4.* perform the final additions corresponding to the overlaps in (16).

This method is detailed in Algorithm 1.

**Algorithm 1** Four way split recombined multiplication

**Require:** $A$ and $B$ two degree $n-1$ polynomials with $n = 2^k \geq 4$.

**Ensure:** $C = A \times B$

**for** $i = 0$ **to** 3 **do**

$\hat{A}_i \leftarrow CPF_{2,n/4}(A_i)$, $\hat{B}_i \leftarrow CPF_{2,n/4}(B_i)$

**end for**

**for** $i = 0$ **to** 3 **do**

**for** $j = 0$ **to** 3 **do**

$\hat{C}_{i,j} \leftarrow CM_{2,n/4}(\hat{A}_i, \hat{B}_j)$

**end for**

**end for**

$\hat{C}_0 \leftarrow \hat{C}_{0,0}$, $\hat{C}_1 \leftarrow CA_{2,n/4}(\hat{C}_{0,1}, \hat{C}_{1,0})$, $\hat{C}_2 \leftarrow CA_{2,n/4}(\hat{C}_{0,2}, \hat{C}_{1,1}, \hat{C}_{2,0})$,
$\hat{C}_3 \leftarrow CA_{2,n/4}(\hat{C}_{0,3}, \hat{C}_{1,2}, \hat{C}_{2,1}, \hat{C}_{3,0})$, $\hat{C}_4 \leftarrow CA_{2,n/4}(\hat{C}_{1,3}, \hat{C}_{2,2}, \hat{C}_{3,1})$,
$\hat{C}_5 \leftarrow CA_{2,n/4}(\hat{C}_{2,3}, \hat{C}_{3,2})$, $\hat{C}_6 \leftarrow \hat{C}_{3,3}$

**for** $i = 0$ **to** 6 **do**

$C_i \leftarrow R(\hat{C}_i)$

**end for**

**return**$(C_0 + C_1 X^{n/4} + C_2 X^{2n/4} + C_3 X^{3n/4} + C_4 X^{4n/4} + C_5 X^{5n/4} + C_6 X^{6n/4})$

We now evaluate the complexity of the architecture corresponding to Algorithm 1. We express the space complexity in terms of the complexities of the different functions $CPF$, $CM$, $CA$ and $R$ used in Algorithm 1 plus the cost of the additions in the final step. These additions in the final step of Algorithm 1 has a cost of $3n/2 - 6$ XOR gates. Therefore, we obtain the following expression for the four-way recombined multiplier

$$\begin{aligned} \mathcal{S}_{\oplus}(n) &= 8\mathcal{S}^{CPF}_{2,\oplus}(n/4) + 7\mathcal{S}^{R}_{2,\oplus}(n/4) + 9\mathcal{S}^{CA}_{2,\oplus}(n/2) + 3n/2 - 6, \\ \mathcal{S}_{\otimes}(n) &= 16\mathcal{S}^{CM}_{2,\otimes}(n/4). \end{aligned}$$

If we now apply the block complexities given in Lemma 1 and Table 5 to the above expressions, we obtain the complexities summarized in Table 7.

In order to evaluate the delay of the multiplier, we note that the critical paths are the paths which go

through $\hat{C}_2$, $\hat{C}_3$ or $\hat{C}_4$, and these paths have the same delay:

$$\mathcal{D}(n) \quad = \quad 3D_{\oplus} + \underbrace{\mathcal{D}_{CPF}(n/4) + \mathcal{D}_{CM}(n/4) + \mathcal{D}_{R}(n/4)}_{(*)}.$$

In the above equation $(*)$ is equal to the delay of a multiplier with input size $n/4$ bits. Using the delay formulas presented in Sections 2 and 3, we obtain the corresponding delays reported in Table 7.

TABLE 7

Complexity of four-way recombined multiplier

| Method | # AND | # XOR | Delay |
|---|---|---|---|
| Original Karatsuba (Subsection 2.1.1) | $\frac{16}{9}n^{\log_2(3)}$ | $5n^{\log_2(3)} - 11 + 8$ | $(3\log_2(n) - 3)D_{\oplus} + D_{\otimes}$ |
| Two-way of [1], [10] (Subsection 2.1.2) | $\frac{16}{9}n^{\log_2(3)}$ | $\frac{83}{18}n^{\log_2(3)} - \frac{37n}{4} + \frac{9}{2}$ | $(3\log_2(n) - 3)D_{\oplus} + D_{\otimes}$ |
| Two-way of [3] (Subsection 2.1.3) | $\frac{16}{9}n^{\log_2(3)}$ | $5n^{\log_2(3)} - 11 + 8$ | $(2\log_2(n) - 1)D_{\oplus} + D_{\otimes}$ |
| Four-way of [1] (Subsection 2.3) | $\frac{16}{9}n^{\log_2(3)}$ | $\frac{1639}{360}n^{\log_2(3)} - \frac{89n}{10} - \frac{29}{8}$ | $(\frac{5}{2}\log_2(n) - 2)D_{\oplus} + D_{\otimes}$ |
| Proposed four-way in Subsection 3.3 | $\frac{16}{9}n^{\log_2(3)}$ | $\frac{353}{72}n^{\log_2(3)} - 11n + \frac{71}{8}$ | $(2\log_2(n) - 1)D_{\oplus} + D_{\otimes}$ |

### 4.5 Three-way split recombination

We now present a block recombination for three-way split multipliers. Let $A$ and $B$ be two polynomials of degree $n-1$ where $n$ is a power of 3. We split $A$ and $B$ in three parts $A = A_0 + A_1X^{n/3} + A_2X^{2n/3}$ and $B = B_0 + B_1X^{n/3} + B_2X^{2n/3}$ where $A_i$ and $B_i$ have degree $n/3 - 1$ for $i = 0, 1, 2$. After expanding the product $AB$, we obtain:

$$AB = \underbrace{A_0B_0}_{C_0} + \underbrace{(A_0B_1 + A_1B_0)}_{C_1}X^{n/3} + \underbrace{(A_0B_2 + A_1B_1 + A_2B_0)}_{C_2}X^{2n/3} + \underbrace{(A_1B_2 + A_2B_1)}_{C_3}X^{3n/3} + \underbrace{A_2B_2}_{C_4}X^{4n/3}. \tag{17}$$

Based on the above expression of $AB$ we can compute the product of $A$ and $B$ as follows. We apply the function $CVF_{3,n/3}$ to $A_i$ and $B_i$ for $i = 0, 1, 2$ and perform the component multiplication $CM_{3,n/3}(A_i, B_j)$ for $i = 0, 1, 2$ and $j = 0, 1, 2$. Then we perform the addition in the expression of $C_1$, $C_2$ and $C_3$ of (17). Finally, we apply the function $R_{3,n/3}$ to $\hat{C}_i, i = 0, \ldots, 4$ and perform the overlapping additions corresponding to (17).

We evaluate the space complexity of the architecture corresponding to Algorithm 2. We add the contributions of the functions $CVF$, $CM$, $CA$ and $R$ appearing in Algorithm 2 plus the final overlapping additions, which gives:

$$\begin{aligned} \mathcal{S}_{\oplus}(n) &= 6\mathcal{S}^{CPF}_{3,\oplus}(n/3) + 5\mathcal{S}^{R}_{3,\oplus}(n/3) + 4\mathcal{S}^{CA}_{3,\oplus}(n/3) + 4n/3 - 4, \\ \mathcal{S}_{\otimes}(n) &= 9\mathcal{S}^{CM}_{3,\otimes}(n/3). \end{aligned}$$

We obtain gate counts as summarized in in Table 8 by using the complexities of the three-way split function $CVF$, $CM$, $CA$ and $R$ given in Lemma 1 and Table 1.

**Algorithm 2** Three-way multiplication using block recombination

**Require:** $A$ and $B$ two degree $n-1$ polynomials with $n = 2^k \geq 4$.

**Ensure:** $C = A \times B$

**for** $i = 0$ **to** 2 **do**

$\hat{A}_i \leftarrow CPF(A_i)$, $\hat{B}_i \leftarrow CPF(B_i)$

**end for**

**for** $i = 0$ **to** 2 **do**

**for** $j = 0$ **to** 2 **do**

$\hat{C}_{i,j} \leftarrow CM(\hat{A}_i, \hat{B}_j)$

**end for**

**end for**

$\hat{C}_0 \leftarrow \hat{C}_{0,0}$, $\hat{C}_1 \leftarrow CA(\hat{C}_{0,1}, \hat{C}_{1,0})$, $\hat{C}_2 \leftarrow CA(\hat{C}_{0,2}, \hat{C}_{1,1}, \hat{C}_{2,0})$, $\hat{C}_3 \leftarrow CA(\hat{C}_{1,2}, \hat{C}_{2,1})$, $\hat{C}_4 \leftarrow \hat{C}_{2,2}$

**for** $i = 0$ **to** 4 **do**

$C_i \leftarrow R(\hat{C}_i)$

**end for**

**return**$(C_0 + C_1 X^{n/3} + C_2 X^{2n/3} + C_3 X^{3n/3} + C_4 X^{4n/3})$

We now evaluate the delay of the architecture based on Algorithm 2. The delay of the critical path of this architecture is as follows

$$\mathcal{D}(n) = \underbrace{\mathcal{D}^{CVF}_{3,n/3} + \mathcal{D}^{CM}_{3,n/3} + \mathcal{D}^{R}_{3,n/3}}_{(*)} + 3D_{\oplus}.$$

In the above expression, $(*)$ represents the delay of a multiplier of size $n/3$. The delay of the three-way recombined multiplier is then derived from the delay of the different three-way split formula given Sections 2 and 3. The resulting complexities are reported in Table 8.

TABLE 8

New space and time complexities for $n = 3^k$

| Method | # AND | # XOR | Delay |
|---|---|---|---|
| Space optimized formula [2] (Subsection 2.2.1) | $\frac{9}{6}n^{\log_3(6)}$ | $\frac{79}{18}n^{\log_2(3)} - \frac{149n}{18} + 3$ | $(4\log_3(n) - 1)D_{\oplus} + D_{\otimes}$ |
| Delay optimized formula [3] (Subsection 2.2.2) | $\frac{9}{6}n^{\log_3(6)}$ | $\frac{40}{9}n^{\log_2(3)} - \frac{169n}{18} + 6$ | $3\log_3(n)D_{\oplus} + D_{\otimes}$ |
| Proposed space optimized formula (Subsection 3.1) | $\frac{9}{6}n^{\log_3(6)}$ | $4n^{\log_3(6)} - \frac{n}{2} + 2$ | $(4\log_3(n) - 1)D_{\oplus} + D_{\otimes}$ |
| Proposed delay optimized formula (Subsection 3.2) | $\frac{9}{6}n^{\log_3(6)}$ | $\frac{13}{3}n^{\log_3(6)} - \frac{19n}{18} + 5$ | $3\log_3(n)D_{\oplus} + D_{\otimes}$ |

## 5 COMPLEXITY COMPARISON AND CONCLUSION

Here we provide comparisons and some concluding remarks. First, in Subsection 5.1, we compare the two-way split approaches considered in this paper. This subsection also includes results for the four-way split approach as the latter is an extension of the two-way approach. The three-way split approaches are compared in Subsection 5.2 followed by some conluding remarks.

### 5.1 Comparison of two-way split approaches

Table 9 summarizes the complexity of the two-way split approaches, i.e., non-recombined and recombined formulas, for the multiplication of polynomials.

TABLE 9

Multiplier space and time complexities for $n = 2^k$

| | Method | # AND | # XOR | Delay |
|---|---|---|---|---|
| Non-recomb. | Two-way of [1], [10] | $n^{\log_2(3)}$ | $5.5n^{\log_2(3)} - 7n + 1.5$ | $D_\otimes + 3\log_2(n)D_\oplus$ |
| | Two-way of [3] | $n^{\log_2(3)}$ | $6n^{\log_2(3)} - 8n + 2$ | $D_\otimes + 2\log_2(n)D_\oplus$ |
| | Four-way split formula of [1] (Section 2.3) | $n^{\log_2(3)}$ | $5.43n^{\log_2(3)} - 6.8n + 1.38$ | $D_\otimes + 2.5\log_2(n)D_\oplus$ |
| | Proposed four-way split formula (Subsection 3.3) | $n^{\log_2(3)}$ | $5.88n^{\log_2(3)} - 8n + 2.13$ | $D_\otimes + 2\log_2(n)D_\oplus$ |
| Recomb. | Four-way split formula of [1] (Subsection 2.3) | $1.77n^{\log_2(3)}$ | $4.56n^{\log_2(3)} - 8.9n + 3.625$ | $D_\otimes + 2.5\log_2(n)D_\oplus$ |
| | Proposed four-way split formula (Subsection 3.3) | $1.77n^{\log_2(3)}$ | $4.91n^{\log_2(3)} - 11n + 8.88$ | $D_\otimes + 2\log_2(n)D_\oplus$ |

If we only consider non-recombined multipliers in Table 9, we see that the four-way approach of [1] has a better space and time complexities compared to the two-way approach of [1], [10]. To the best of our knowledge this four-way approach in terms of the overall space complexity (AND and XOR gates combined). In terms of the time complexity, the two-way approach of [3] offers the best result among thee existing non-recombined multipliers. Our proposed four-way formula achieves the same time complexity of [3] but with fewer gates. If we consider the space-time product complexity, the proposed four-way formula has the best result among all non-recombined two-way split formulas.

If we now compare all recombined and non-recombined approaches listed in Table 9, we see that the best approach, when we consider only the overall space complexity, is the recombined four-way formula of [1], which requires $6.33n^{\log_2(3)} + O(n)$ gates operations. On the other hand, the recombined four-way split formula of Subsection 3.3 matches the best timing result of [3] with slightly more gates, but has the least area-time product among all the methods listed in Table 9.

### 5.2 Comparison of three-way split approaches

In Table 10 we give the complexity of polynomial multiplication based on three-way split methods.

Let us fist compare the non-recombined formulas. Based on the results in Table 10, we remark that the proposed method in Subsection 3.1 has the smallest space requirement. The proposed delay efficient

TABLE 10

Multiplier space and time complexities for $n = 3^k$

| | Method | # AND | # XOR | Delay |
|---|---|---|---|---|
| Non-recomb. | Space efficient multiplier [2] | $n^{\log_3(6)}$ | $5.27n^{\log_3(6)} - 6.67n + 1.4$ | $D_\otimes + 4\log_3(n)D_\oplus$ |
| | Delay efficient multiplier [3] | $n^{\log_3(6)}$ | $5.33n^{\log_3(6)} - 7.33n + 2$ | $D_\otimes + 3\log_3(n)D_\oplus$ |
| | Proposed space efficient method in Subsection 3.1 | $n^{\log_3(6)}$ | $4.8n^{\log_3(6)} - 6n + 1.2$ | $D_\otimes + 4\log_3(n)D_\oplus$ |
| | Proposed delay efficient method in Subsection 3.1 | $n^{\log_3(6)}$ | $5.20n^{\log_3(6)} - 7n + 1.8$ | $D_\otimes + 3\log_3(n)D_\oplus$ |
| Recomb. | Recombined space efficient multiplier of Subsection 3.1 | $1.5n^{\log_3(6)}$ | $4n^{\log_3(6)} - 0.5n + 2$ | $D_\otimes + (4\log_3(n) - 1)D_\oplus$ |
| | Recombined space efficient multiplier of Subsection 3.2 | $1.5n^{\log_3(6)}$ | $4.33n^{\log_3(6)} - 1.05n + 5$ | $D_\otimes + 3\log_3(n)D_\oplus$ |

method in Subsection 3.1 matches the least time complexity result of the three-way split formula of [3] with few number of gates.

If we now consider the recombined formulas, we see that the recombination to the proposed two methods (Subsection 3.1 and Subsection 3.2) reduce their total gate counts without increasing their delays.

### 5.3 Concluding remarks

Multiplication of binary polynomials using sub-quadratic arithmetic complexity is often used in today's cryptographic systems, such as those based on elliptic curves and pairing [7], [4]. To this end, in this paper we have considered efficient bit parallel designs of sub-quadratic space complexity polynomial multipliers. Specifically, we have proposed improvements in terms of gate counts and delay to the existing best two- and three-way multipliers. The improvements have been achieved by identifying and removing some re-occurring computations. We have also proposed improvements to Bernstein's formula for the four-way split multiplication. More importantly, for the first time ever we have applied the block recombination approach to polynomial multiplication. This has provided a new area-time trade-offs and the least area-time product complexity among various multiplication schemes considered in the paper.

## REFERENCES


[1] D. J. Bernstein. Batch Binary Edwards. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 317–336, 2009.

[2] M. Cenk, C. Negre, and M. A. Hasan. Improved Three-Way Split Formulas for Binary Polynomial and Toeplitz Matrix Vector Products. *IEEE Trans. Comp.*, to appear.

[3] H. Fan, J. Sun, M. Gu, and K.-Y. Lam. Overlap-free Karatsuba-Ofman polynomial multiplication algorithms. *Information Security, IET*, 4:8–14, march 2010.

[4] S. Galbraith. Pairings. In *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005.

[5] M. A. Hasan, N. Méloni, A. Namin, and C. Negre. Block Recombination Approach for Subquadratic Space Complexity Binary Field Multiplication Based on Toeplitz Matrix-Vector Product. *IEEE Trans. Computers*, 61(2):151–163, 2012.

[6] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[7] V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology, proceeding's of CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer-Verlag, 1986.

[8] C. Paar. A new architecture for a parallel finite field multiplier with low complexity based on composite fields. *IEEE Trans. Comput.*, 45(7):856–861, 1996.

[9] B. Sunar. A Generalized Method for Constructing Subquadratic Complexity GF($2^k$) Multipliers. *IEEE Transactions on Computers*, 53:1097–1105, 2004.

[10] G. Zhou and H. Michalik. Comments on "A New Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Field". *IEEE Trans. Computers*, 59(7):1007–1008, 2010.