Efficient Binary Polynomial Multiplication Based on Optimized Karatsuba Reconstruction

Christophe Negre

Efficient Binary Polynomial Multiplication Based on Optimized Karatsuba Reconstruction

C. Negre

Abstract

At Crypto 2009 [2], Bernstein proposed two optimized Karatsuba formulas for binary polynomial multiplication. Bernstein obtained these optimizations by re-expressing the reconstruction of one or two recursions of the Karatsuba formula. In this paper we present a generalization of these optimizations. Specifically, we optimize the reconstruction of s recursions of the Karatsuba formula for $s \geq 1$. To reach this goal, we express the recursive reconstruction through a tree and re-organize this tree to derive an optimized recursive reconstruction of depth s. When we apply this approach to a recursion of depth $s = \log_2(n)$ we obtain a parallel multiplier with a space complexity of $5.25n^{\log_2(3)} + O(n)$ XOR gates and $n^{\log_2(3)}$ AND gates and with a delay of $2\log_2(n)D_{\oplus} + D_{\otimes}$.

Index Terms

Binary polynomial multiplication, Karatsuba formula, optimized recursive reconstruction.

1 INTRODUCTION

Finite fields are intensively used in cryptographic applications [9], [7], [3], [4] and in error control coding [1]. Specifically, a single cryptographic protocol based on elliptic curve requires several hundreds additions and multiplications in a finite field of size $[2^{160}, 2^{600}]$. An addition in a finite field is generally simpler to implement than a multiplication which remains quite costly in time and resources. Consequently, during the past two decades, an important amount of work have been done to improve the efficiency of the multiplication in finite field.

In this paper we focus on hardware design of parallel multiplier for extended binary fields $\mathbb{F}_{2^n}$. A multiplication in such fields consists of a multiplication of binary polynomial followed by a reduction modulo an irreducible polynomial P of degree n. The most costly part of the finite field multiplication is the polynomial multiplication, since, when P is sparse (i.e., when P is a trinomial or a pentanomial), the reduction has a cost of $O(n)$ bit additions. A binary polynomial multiplication can been done through

a quadratic circuit [8] which requires $O(n^2)$ bit operations and is logarithmic in time $\log_2(n)D_\oplus + D_\otimes$ where $D_\otimes$ and $D_\oplus$ are the delay of an AND gate and an XOR gate respectively. This type of multiplier is the fastest among all kind of multipliers, but due to their quadratic complexity, they are costly in space for the considered field size $160 \leq n \leq 500$. In order to reduce the space requirement, subquadratic space complexity multiplier based on the Karatsuba approach [6] have been proposed in [10]. A multiplier based on the Karatsuba formula has a space complexity of $6n^{\log_2(3)} + O(n)$ XOR gates and $n^{\log_2(3)}$ AND gates with a delay $3\log_2(n)D_\oplus + D_\otimes$.

Recently some optimizations have been proposed to reduce the space complexity or the delay of the Karatsuba multiplier. The first improvement was proposed by Fan *et al.* in [5]: they use a different splitting. Their method reduces the delay of the Karatsuba multiplier to $2\log_2(n)D_\oplus + D_\otimes$ but the space complexity remains the same. The second improvement was proposed by Berstein [2]: Bernstein optimizes the reconstruction part of the Karatsuba formula by factorizing some constant common terms. Bernstein applied this optimization to the reconstruction of Karatsuba formula and then to two recursions of Karatsuba resulting in $5.46n^{\log_2(n)} + O(1)$ XOR gates instead of $6n^{\log_2(n)} + O(n)$ XOR gates for the original Karatsuba formula and a delay of $2.5\log_2(n)D_\oplus + D_\otimes$.

We propose to extend the idea of Bernstein to s recursions of the Karatsuba formula. We obtain this extension by first expressing the s recursions of the Karatsuba reconstruction in a tree structure. We then re-organize the operations in this tree leading to the optimization of the s recursions of the Karatsuba reconstruction. We also give an algorithmic form of the resulting reconstruction and evaluate thoroughly the complexity of this approach. We will show that if we apply this method to a recursion of depth $s = \log_2(n)$, we obtain a parallel multiplier with a space complexity of $5.25n^{\log_2(3)} + O(n)$ XOR gates and $n^{\log_2(3)}$ AND gates and a delay of $2\log_2(n)D_\oplus + D_\otimes$.

The paper is organized as follows: in Section 2, we review the original Karatsuba formula and the optimized versions proposed by Bernstein [2]. In Section 3 we generalize the approach of Bernstein to s recursions of Karatsuba formula. Finally, we compare the complexities of the proposed method with best known approaches and give some concluding remarks in Section 4.

2 Review of Karatsuba formula

We review in this section the method of Karatsuba for the multiplication of binary polynomials. We also review the recent optimizations presented by Bernstein at Crypto 2009 [2].

2.1 Karatsuba formula

We consider two degree $n-1$ polynomials $A(X) = \sum_{i=0}^{n-1} a_i X^i$ and $B(X) = \sum_{i=0}^{n-1} b_i X^i$ in $\mathbb{F}_2[X]$ with $n = 2^k$. The method of Karatsuba for polynomial multiplication consists of expressing the product $C = A \times B$ in terms of three multiplications of polynomial of half size. The detailed computations are given below:

- *Component polynomial formation (CPF).* The component polynomial formation consists of splitting A in two halves $A(X) = \underbrace{\sum_{i=0}^{n/2-1} a_i X^i}_{A_L} + X^{n/2} \underbrace{\sum_{i=0}^{n/2-1} a_{i+n/2} X^i}_{A_H}$ and then generate three polynomials of half size $A'_0 = A_L, A'_1 = A_L + A_H$ and $A'_2 = A_H$. The same is done for $B = B_L + B_H X^{n/2}$: we generate $B'_0 = B_L, B'_1 = B_L + B_H$ and $B'_2 = B_H$.
- *Recursive products.* We perform the pairwise products of the CPF of A and B

$$\begin{aligned} C'_0 &= A'_0 B'_0 = A_L B_L, \\ C'_1 &= A'_1 B'_1 = (A_L + A_H)(B_L + B_H), \\ C'_2 &= A'_2 B'_2 = A_H B_H. \end{aligned} \tag{1}$$

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

$$C = C'_0(1 + X^{n/2}) + C'_1 X^{n/2} + C'_2 X^{n/2}(1 + X^{n/2}), \tag{2}$$

$$= C'_0 + (C'_0 + C'_1 + C'_2) X^{n/2} + C'_2 X^n. \tag{3}$$

The three half size products C'_0, C'_1 and C'_2 of (1) are computed by applying the same method recursively. If the recursive computations are performed in parallel we get a parallel multiplier with a subquadratic space complexity and a logarithmic delay. Specifically, the number of XOR gates ($\mathcal{S}_\oplus(n)$), AND gates ($\mathcal{S}_\otimes(n)$) and the delay are given in (4) is given in a recursive form (left) and a non-recursive form (right).

$$\begin{cases} \mathcal{S}_\oplus(n) = 4n - 4 + 3\mathcal{S}_\oplus(n), \\ \mathcal{S}_\otimes(n) = 3\mathcal{S}_\otimes(n), \\ \mathcal{D}(n) = 3D_\oplus + \mathcal{D}(n/2). \end{cases} \qquad \begin{cases} \mathcal{S}_\oplus(n) = 6n^{\log_2(3)} - 8n + 2, \\ \mathcal{S}_\otimes(n) = n^{\log_2(3)}, \\ \mathcal{D}(n) = 3\log_2(n) D_\oplus + D_\otimes. \end{cases} \tag{4}$$

In the previous equation, $D_\oplus$ represents the delay of an XOR gate and $D_\otimes$ the delay of an AND gate.

Optimization of the reconstruction. Recently an optimized version of the Karatsuba formula have been proposed: Bernstein in [2] have reduced the complexity of the reconstruction step as follows

$$\begin{aligned} &\textit{Step 1.} \quad R_0 = P_0 + X^{n/2} P_1, && (\text{Cost } = n/2 - 1 \text{ bit additions}), \\ &\textit{Step 2.} \quad R_1 = R_0(1 + X^{n/2}), && (\text{Cost } = n - 1 \text{ bit additions}), \\ &\textit{Step 3.} \quad C = R_1 + P_2 X^{n/2}, && (\text{Cost } = n - 1 \text{ bit additions}). \end{aligned} \tag{5}$$

This approach reduces the number of bit additions of one recursion of the Karatsuba formula $\mathcal{S}_\oplus(n) = 7n/2 - 3 + 3\mathcal{S}_\oplus(n/2)$ which gives for a full recursion $\mathcal{S}_\oplus(n) = 5.5n^{\log_2(n)} - 7n + 3/2$. But this approach conserves a delay of $\mathcal{D}(n) = 3\log_2(n) D_\oplus + D_\otimes$. In the sequel we will call the reconstruction formula (5) as Bernstein's Reconstruction (BR_1) of one recursion of the Karatsuba formula.

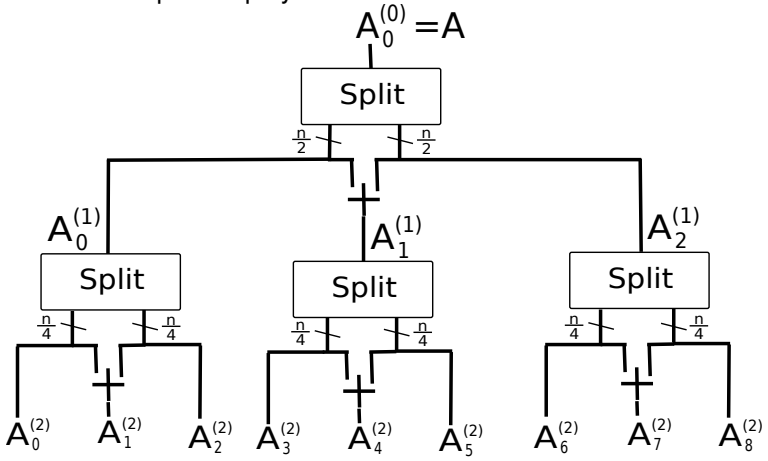
Remark 1. The optimization of the reconstruction of the Karatsuba formula was also proposed by Zhou and Michalik in [11], independently from Bernstein. We have presented the approach of Bernstein since his approach is the base of our generalization.

2.2 Bernstein's optimization of two recursions of Karatsuba formula

In [2] Bernstein has also extended his optimization to two recursions of Karatsuba formula. We here present this approach in a slightly different manner.

Component formation and recursive products. The component formation is recursively applied twice resulting in 9 terms of size $n/4$. One recursion consists of a splitting in two halves and an addition of these two halves. The corresponding circuit is depicted in Fig. 1. If we split $A = A_{LL} + A_{LH}X^{n/4} +$

Fig. 1. Two recursions of the component polynomial formation



$A_{HL}X^{2n/4} + A_{HH}X^{3n/4}$, the nine terms of size $n/4$ generated by the two recursions of the CPF are as follows

$$\begin{array}{lll} A_0^{(2)} = A_{LL}, & A_1^{(2)} = (A_{LL} + A_{LH}), & A_1^{(2)} = A_{LH}, \\ A_3^{(2)} = (A_{LL} + A_{HL}), & A_4^{(2)} = (A_{LL} + A_{LH} + A_{HL} + A_{HH}), & A_5^{(2)} = (A_{LH} + A_{HH}), \\ A_6^{(2)} = A_{HL}, & A_7^{(2)} = (A_{HL} + A_{HH}), & A_8^{(2)} = A_{HH}. \end{array}$$

The same formula is applied to B which results in nine terms $B_i^{(2)}$, $i = 0, 1, \ldots, 8$ of size $n/4$. The nine recursive products are $C_i^{(2)} = A_i^{(2)} \times B_i^{(2)}, i = 0, 1 \ldots, 8$ and have degree $2n/4 - 2$.

Reconstruction. The first recursion in the reconstruction process produces $C_i^{(1)}, i = 0, 1, 2$ in terms of $C_i^{(2)}, i = 0, 1, \ldots, 8$. Specifically, $C_0^{(1)}$ is expressed in terms of $C_i^{(2)}, i = 0, 1, 2$, and $C_1^{(1)}$ is expressed in terms of $C_i^{(2)}, i = 3, 4, 5$, and $C_2^{(1)}$ is expressed in terms of $C_i^{(2)}, i = 6, 7, 8$, as follows:

$$\begin{array}{rcl} C_0^{(1)} & = & (1 + X^{n/4})C_0^{(2)} + X^{n/4}C_1^{(2)} + X^{n/4}(1 + X^{n/4})C_2^{(2)}, \\ C_1^{(1)} & = & (1 + X^{n/4})C_3^{(2)} + X^{n/4}C_4^{(2)} + X^{n/4}(1 + X^{n/4})C_5^{(2)}, \\ C_2^{(1)} & = & (1 + X^{n/4})C_6^{(2)} + X^{n/4}C_7^{(2)} + X^{n/4}(1 + X^{n/4})C_8^{(2)}. \end{array} \quad (6)$$

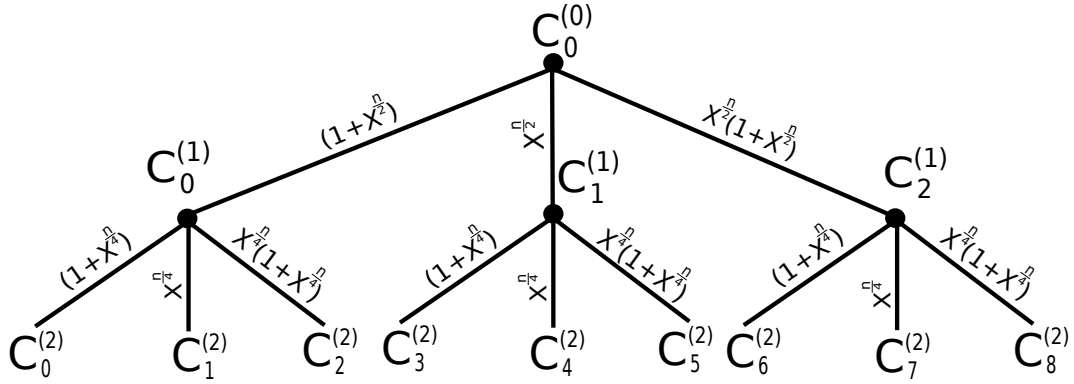
We now apply a second recursion of the reconstruction process and we obtain $C = C^{(0)}$ in terms of $C_0^{(1)}, C_1^{(1)}, C_2^{(1)}$ as

$$C = (1 + X^{n/2})C_0^{(1)} + X^{n/2}C_1^{(1)} + X^{n/2}(1 + X^{n/2})C_2^{(1)}. \quad (7)$$

Note that the superscript (i) of $A^{(i)}, B^{(i)}$ and $C^{(i)}$ indicates the depth of the data in the recursion.

The operations of (6) and (7) can be organized in a tree structure: starting from the root $C = C_0^{(0)}$ which is linked to the three children $C_0^{(1)}, C_1^{(1)}$ and $C_2^{(1)}$. The links are labelled by the respective factors $(1 + X^{n/2}), X^{n/2}$ and $X^{n/2}(1 + X^{n/2})$ of $C_i^{(1)}, i = 0, 1, 2$, appearing in (7). We then repeat this process for each $C_i^{(1)}, i = 0, 1, 2$. The resulting reconstruction tree is shown in Fig. 2

Fig. 2. Reconstruction tree of two recursions of Karatsuba

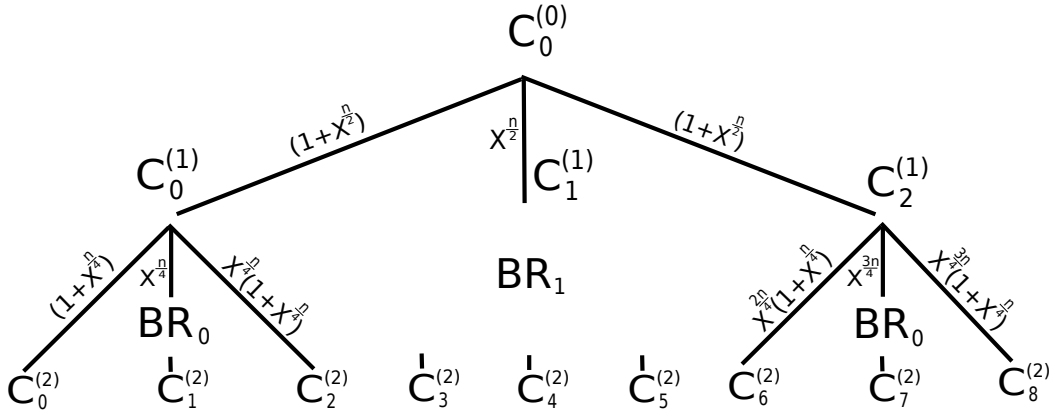


We now modify this tree by performing the following changes:

- We first replace the sub-tree below $C_1^{(1)}$ by a block representing the reconstruction formula BR_1 given in (5) in terms of $C_3^{(2)}, C_4^{(2)}, C_5^{(2)}$.
- We replace also the middle terms $C_1^{(2)}$ and $C_7^{(2)}$ by a block BR_0 representing a reconstruction of depth 0: by convention $BR_0(U) = U$ for any U.
- We then move down the factor $X^{n/2}$, which appears in the label of the link between $C_0^{(0)}$ and $C_2^{(1)}$, to the three links below $C_2^{(1)}$.

The resulting modified tree is shown in Fig. 3. This tree provides Bernstein's reconstruction for two

Fig. 3. Modified reconstruction tree of two recursions of the Karatsuba formula



recursions of the Karatsuba formula. We treat each operation labelled by the different link, starting from the bottom and climbing up to the root. We have three steps:

- **Step 1.** We add the leaf-coefficients $C_0^{(2)}, C_2^{(2)} C_6^{(2)}, C_8^{(2)}$ of the tree in Fig. 3 multiplied by their respective factor $1, X^{n/4}, X^{2n/4}$ and $X^{3n/4}$:

$$C \leftarrow C_0^{(2)} + C_2^{(2)} X^{n/4} + C_6^{(2)} X^{2n/4} + C_8^{(2)} X^{3n/4}, \text{ Cost } = 3n/4 - 3.$$

- **Step 2.** We multiply by the common factor $(1 + X^{n/4})$ and add the middle terms of depth two $BR_0(C_1^{(2)})$ and $BR_0(C_7^{(2)})$ multiplied by their respective monomial factor $X^{n/4}$ and $X^{2n/4}$

$$\begin{aligned} C &\leftarrow C(1 + X^{n/4}), \text{ Cost } = n - 1, \\ C &\leftarrow C + X^{n/4} BR_0(C_1^{(2)}) + X^{3n/4} BR_0(C_7^{(2)}), \text{ Cost } = 2(2n/4 - 1). \end{aligned}$$

- **Step 3.** We multiply by $(1+X^{n/2})$ and add the middle term of depth one $C_1^{(1)} = BR_1(C_3^{(2)}, C_4^{(2)}, C_5^{(2)})$ multiplied by its label $X^{n/2}$

$$\left\{ \begin{array}{rcl} C & \leftarrow & C(1 + X^{n/2}), \text{ Cost } = n - 1, \\ C & \leftarrow & C + X^{n/2} BR_1(C_3^{(2)}, C_4^{(2)}, C_5^{(2)}), \text{ Cost } = (2n/2 - 1) + \underbrace{5n/4 - 3}_{BR_1}. \end{array} \right.$$

TABLE 1

Complexity evaluation of Bernstein's optimization of two recursions of Karatsuba

Operation	Computations	# XOR	# AND
CPF	CPF of A	$n/2 + 3n/4$	0
	CPF of B	$n/2 + 3n/4$	0
Rec. Products.	$A_i^{(2)} \cdot B_i^{(2)}, i = 0, 1, \ldots, 8$	$9\mathcal{S}_\oplus(n/4)$	$9\mathcal{S}_\otimes(n/4)$
Reconstruction	Step 1.	$3n/4 - 3$	0
	Step 2	$2n - 3$	0
	Step 3	$13n/4 - 5$	0
	Total	$17n/2 - 11 + 9\mathcal{S}_\oplus(n/4)$	$9\mathcal{S}_\otimes(n/4)$

The non-recursive expression of this optimized approach, when n a power of 4, is as follows

$$\left\{ \begin{array}{rcl} S_\oplus(n) & = & \frac{217}{40} n^{\log_2(3)} - \frac{34n}{5} + \frac{11}{8}, \\ S_\otimes(n) & = & n^{\log_2(3)}. \end{array} \right. \tag{8}$$

The recursive form of the delay of the optimized two recursions of Karatsuba formula is equal to $\mathcal{D}(n) = 5D_\oplus + \mathcal{D}(n/4)$. The resulting non-recursive form of the delay is $\mathcal{D}(n) = \frac{5}{2}\log_2(n) D_\oplus + D_\otimes$ when n is an even power of 2.

3 GENERALIZATION OF BERNSTEIN'S OPTIMIZATION OF KARATSUBA

In this section, we present an optimization of the Karatsuba multiplication which generalizes the approach of Bernstein. We will assume that the recursion in Karatsuba multiplication is performed with a depth s and we will optimize the reconstruction part of the recursions of the Karatsuba formula. For this, we will use a reconstruction tree to facilitate the generalization of Bernstein's optimization.

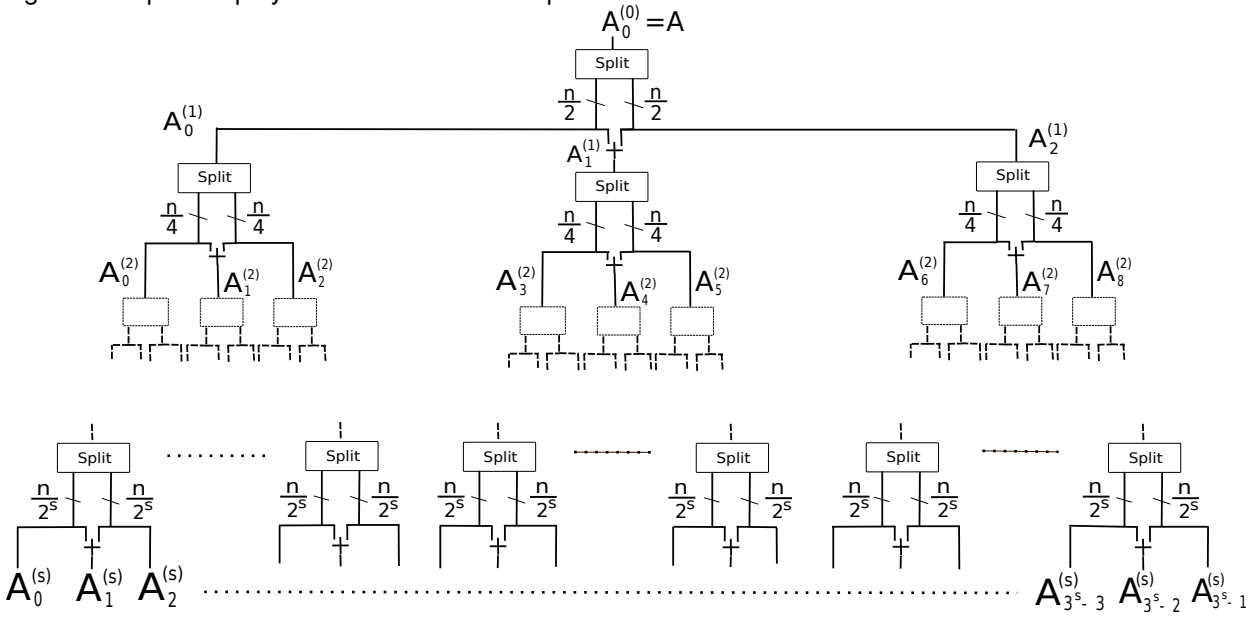
3.1 Karatsuba reconstruction tree of depth s

We split the Karatsuba recursion of depth s into two main steps: a first step which performs component polynomial formations and recursive products and a second step which performs the reconstruction of the product.

Component polynomial formations and recursive products. The component polynomial formation (CPF) consists of recursively splitting in two halves and then generating three polynomials of half size: the two halves and their sum. If we perform the recursion up to a depth s we obtain the circuit shown in Fig. 4. Note that the exponent (h) in the intermediate value $A_i^{(h)}$ represents the depth of $A_i^{(h)}$ in the recursion.

The application of the recursive CPF of depth s to the polynomial A of size n results in 3^s polynomials $A_i^{(s)}, i = 0, 1, \ldots, 3^s - 1$ of size $n/2^s$. Similarly, if we also apply the CPF of depth s to B we obtain 3^s terms $B_i^{(s)}$, $i = 0, 1, \ldots, 3^s - 1$. Then there are 3^s recursive products $C_i^{(s)} = A_i^{(s)} \times B_i^{(s)}, i = 0, \ldots, 3^s - 1$ and the resulting polynomials $C_i^{(s)}$ are of degree $2n/2^s - 2$.

Fig. 4. Component polynomial formation of depth s



Reconstruction tree. The reconstruction consists of applying the formula (2) to each group of three

consecutive $C_i^{(s)}$

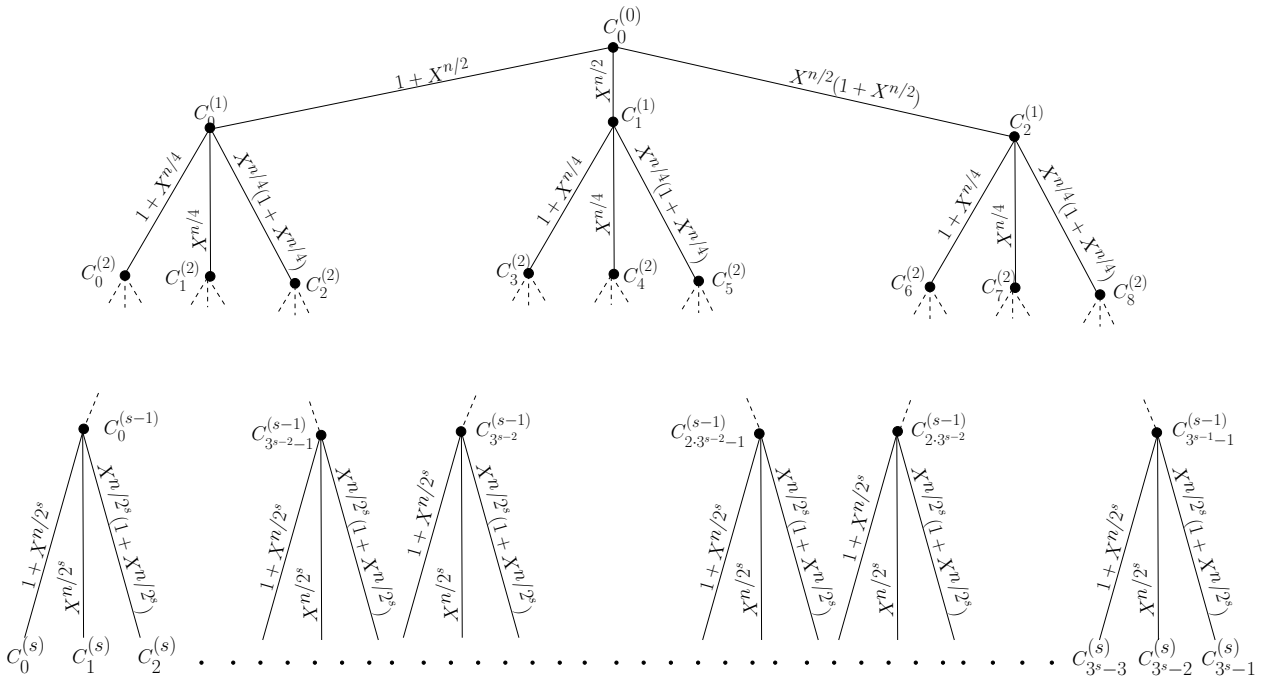
$$C_i^{(s-1)} = C_{3i}^{(s)}(1 + X^{n/2}) + C_{3i+1}^{(s)} X^{n/2} + C_{3i+2}^{(s)} X^{n/2}(1 + X^{n/2}), \tag{9}$$

where $0 \leq i \leq 3^{s-1} - 1$. Then we repeat this process for $s-2, s-1, \ldots, 0$ until we obtain $C = C_0^{(0)}$. These recursive reconstructions can be arranged through a reconstruction tree of depth s which has the following properties:

- The root node is labelled as $C_0^{(0)}$ and corresponds to the reconstructed product $C = A \times B$.
- The intermediate nodes of depth h are labelled as $C_i^{(h)}$ where $0 \leq i < 3^h$. We measure the depth h relatively to the root $C_0^{(0)}$ of the reconstruction tree.
- Each node $C_i^{(h)}$ of depth h is linked down to three children $C_{3i}^{(h+1)}, C_{3i+1}^{(h+1)}$ and $C_{3i+2}^{(h+1)}$ of depth $h+1$. These three links are labelled by $(1 + X^{n/2^{h+1}}), X^{n/2^{h+1}}$ and $X^{n/2^{h+1}}(1 + X^{n/2^{h+1}})$ respectively. This corresponds to one application of the reconstruction formula (2). In the sequel, the link joining $C_i^{(h)}$ to $C_{3i}^{(h+1)}$ will be called the left (L) link starting in $C_i^{(h)}$, the link joining $C_i^{(h)}$ to $C_{3i+2}^{(h+1)}$ will be called the right (R) link and the link $C_i^{(h)}$ to $C_{3i+1}^{(h+1)}$ will be called the middle link (M).

The reconstruction tree of depth s is shown in Fig. 5

Fig. 5. Reconstruction tree of depth s



3.2 The modified reconstruction tree and the generalization of Bernstein's reconstruction.

The first step to get a generalization of Bernstein's reconstruction is to modify reconstruction tree of depth s. But, before proceeding we need to set up some results. We will consider the specific nodes in

the reconstruction tree of depth s defined as follows.

Definition 1. *Let $C_i^{(h)}$ be a node of depth h of the reconstruction tree of depth s, then*

(i) We say that $C_i^{(h)}$ is an L/R node of depth h if the path which starts at the root node and goes to $C_i^{(h)}$ uses only left (L) or right (R) link at each depth.

(ii) We call an L/R-then-M node of depth h a node $C_i^{(h)}$ such that the path which starts from the root node and goes down to $C_i^{(h)}$ uses only left or right link at each depth unless the last link between the middle term $C_i^{(h)}$ and his father.

The following lemma specifies the subscripts j corresponding to the L/R nodes $C_j^{(h)}$.

Lemma 1. *Let h be an integer such that $0 \leq h \leq s$. Let $i = (i_{h-1}, \ldots, i_0)_2$ be the binary representation of an integer i in $[0, 2^h - 1]$, we denote $\sigma(i)$ the integer in $[0, 3^s - 1]$ given by the following base 3 representation*

$$\sigma(i) = (2i_{h-1}, \ldots, 2i_0)_3. \tag{10}$$

We consider a node $C_j^{(h)}$ of depth h in the reconstruction tree of depth s. Then, the node $C_j^{(h)}$ is an L/R node of depth h if and only if there exists i in $[0, 2^h - 1]$ such that $j = \sigma(i)$.

Proof: We prove it by induction on h. For $h = 0$ the unique L/R node is $C_0^{(0)}$ which clearly satisfies the statement of the lemma. We now assume that the statement is true for h and we prove it for $h+1$. Let $C_j^{(h+1)}$ be an L/R node of the reconstruction tree. By definition of an L/R node, all the ancestors of $C_j^{(h+1)}$ are also L/R nodes and in particular $C_k^{(h)}$ the parent of $C_j^{(h+1)}$. By induction hypothesis, there exists $i \in [0, 2^h - 1]$ such that $k = \sigma(i)$. In other words we have $i = (i_{h-1}, \ldots, i_1, i_0)_2$ and $k = (2 \cdot i_{h-1}, \ldots, 2 \cdot i_1, 2 \cdot i_0)_3$. Now, since $C_j^{(h+1)}$ is an L/R node and since it is a child of $C_k^{(h)}$ we have $j = 3k + j_0$ where $j_0 \in \{0, 2\}$. Then, we can deduce from the base 3 representation of k, that

$$\begin{aligned} j &= (2 \cdot i_{h-1}, \ldots, 2 \cdot i_1, 2 \cdot i_0, j_0)_3 \\ &= (2 \cdot i_{h-1}, \ldots, 2 \cdot i_1, 2 \cdot i_0, 2(j_0/2))_3 \end{aligned}$$

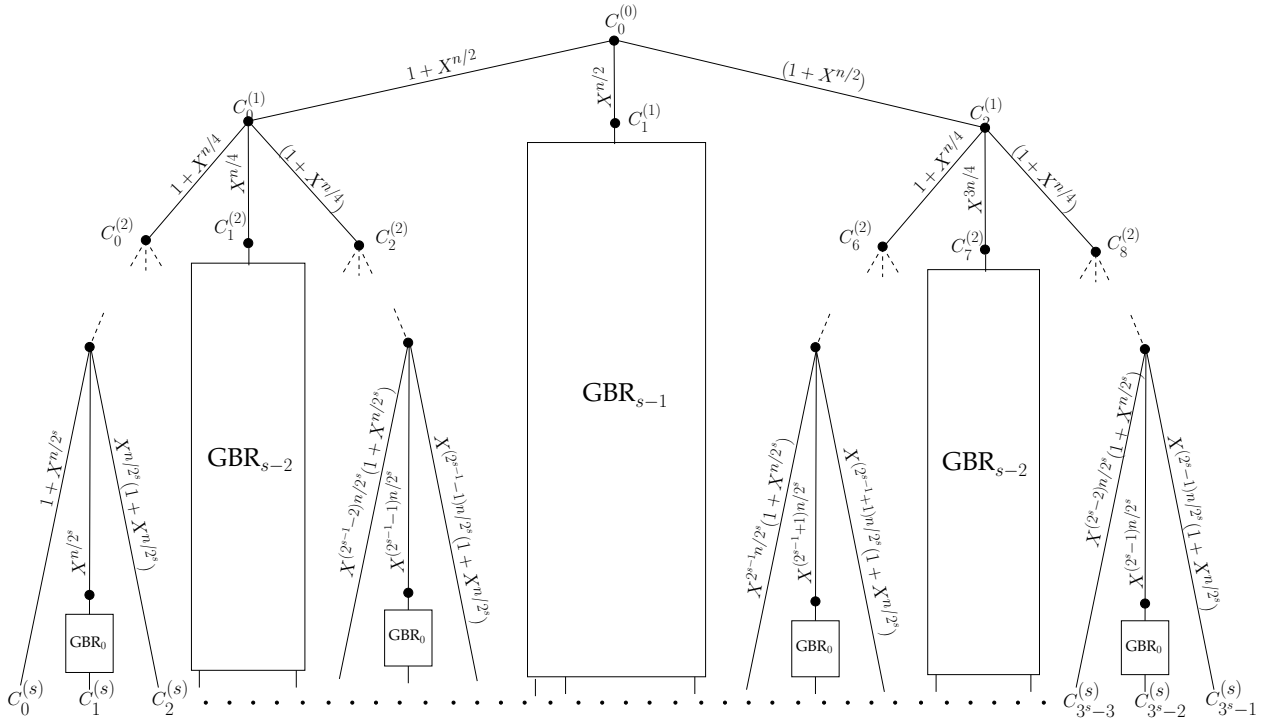
This means that $j = \sigma(i')$ where $i' = (i_{h-1}, \ldots, i_1, i_0, j_0/2)_2$ is in $[0, 2^{h+1} - 1]$. And this concludes the proof. □

We now generalize the modifications done in Fig. 3 to the reconstruction tree of depth 2 . We perform the following modifications:

1) For each L/R-then-M node $C_i^{(h)}$ of depth h we replace its sub-tree by a block labelled GBR$_{s-h}$. This block GBR$_{s-h}$ represents a recursive call to the generalized Bernstein's reconstruction method applied to a smaller tree. The inputs of this block are the leaves $C_{3^{s-h}i}^{(s)}, C_{3^{s-h}i+1}^{(s)}, \ldots, C_{3^{s-h}i+3^{s-h}-1}^{(s)}$ which were the leaves of the replaced sub-tree.
2) We move down, as low as possible, all the factors of type $X^{n/2^h}$ appearing in the link labels. A factor $X^{n/2^h}$ on a link between $C_j^{(h)}$ and his child $C_{3j+2}^{(h+1)}$ is moved down to the three link relying $C_{3j+2}^{(h+1)}$ to his children. We repeat this until we reach either L/R-then-M nodes or L/R leaves.

The modified reconstruction tree of depth s is depicted in Fig. 6.

Fig. 6. Modified reconstruction tree of depth s



Remark 2. We could prove that the link label of the modified tree are as follows

(i) If the link joins up an L/R leave $C_j^{(s)}$ to his parent then there exists $i \in [0, 2^s[$ such that $j = \sigma(i)$ and the link label is $X^{in/2^s}(1 + X^{n/2^s})$.

(ii) If the link joins up an L/R-then-M node $C_j^{(h)}$ to this parent, then there exists $i \in [0, 2^{h-1}[$ such that $j = 3\sigma(i) + 1$ and the link label is $X^{n/2^h} X^{in/2^{h-1}}$.

(iii) Let $C_j^{(h)}$ be a node which is not an L/R leave neither an L/R-then-M node. Then the link which joins up $C_j^{(h)}$ to his parent has $(1 + X^{n/2^h})$ as label.

Since we do not need these three points to be proven in the sequel, we skip this proof here.

3.3 The GBR algorithm

We now use the modified reconstruction tree to derive a generalized Bernstein's reconstruction (GBR). We generalize the scheme used for the modified reconstruction tree of depth 2 given in Subsection 2.2. We accumulate the values of the modified reconstruction tree, depth by depth, starting from the bottom of the tree. These steps are as follows:

- Step 1: Initialization. We accumulate, in an intermediate value U, the values in the L/R leaves $C_{\sigma(i)}^{(s)}$, $i = 0, \ldots, 2^s - 1$, multiplied by their respective factors $X^{\frac{in}{2^s}}$.

- A loop on the depth h starting from s and going down to 1, where we perform the three following operations:
  - Step 2: Multiplication by the depth factor: we multiply U by $(1+X^{2^h})$.
  - Step 3: Reconstruction of the L/R-then-M terms of depth h. We reconstruct the L/R-then-M terms $C^{(h)}_{3\sigma(i)+1}, i=0,\ldots,2^{h-1}-1$, of depth h by applying GBR_{s-h} to the leaves of each sub-tree with root $C^{(h)}_{3\sigma(i)+1}$.
  - Step 4: Accumulation of the reconstructed L/R-then-M terms. We accumulate into U the values of the reconstructed L/R-then-M nodes $C^{(h)}_{3\sigma(i)+1}$ multiplied by their respective factors $X^{\frac{n}{2^h}} \cdot X^{\frac{in}{2^{h-1}}}$.

The above method is specified in Algorithm 1.

Algorithm 1 GBR_s

Require: $C^{(s)}_0,\ldots,C^{(s)}_{3^s-1}$
Ensure: $U=\mathrm{GBR}_{2^s}(C_0,\ldots,C_{3^s-1})$
$U \leftarrow (\sum_{i=0}^{2^s-1} C^{(s)}_{\sigma(i)} X^{\frac{in}{2^s}})$ *//Step 1: Initialization*
for $h=s$ **to** 1
  $U \leftarrow U \times (1+X^{\frac{n}{2^h}})$ *//Step 2: Multiplication of depth h*
  for $i=0$ **to** $2^{h-1}-1$ } *//Step 3 : Reconstruction of the L/R-then-M terms*
    $j \leftarrow 3\sigma(i)+1$
    $V_i \leftarrow \mathrm{GBR}_{s-h}(C^{(s)}_{3^{s-h}j},\ldots,C^{(s)}_{3^{s-h}j+3^{s-h}-1})$
  end for
  $U \leftarrow U + X^{\frac{n}{2^h}} \left(\sum_{i=0}^{2^{h-1}-1} V_i X^{\frac{in}{2^{h-1}}}\right)$ *//Step 4 : Accumulation in U of the L/R-then-M terms*
end for
return(U)

The following theorem establishes the validity of Algorithm 1, i.e., that it correctly reconstructs the polynomial C.

Theorem 1 (Validity of Algorithm 1)**.** *Let us consider two degree n polynomials A and B where $n=2^s n'$ and let $A^{(s)}_0,\ldots,A^{(s)}_{3^s-1}$ and $B^{(s)}_0,\ldots,B^{(s)}_{3^s-1}$ be their respective component polynomial formation of depth s. Then if $C^{(s)}_i = A^{(s)}_i \times B^{(s)}_i$, $i=0,\ldots,3^s-1$ are their pairwise products and if Algorithm 1 is run with inputs $C^{(s)}_0,\ldots,C^{(s)}_{3^s-1}$, it outputs a polynomial U which satisfies $U=A\times B$.*

Proof: We prove the theorem by induction on s. First, since for $s=1$ and $s=2$ Algorithm 1 corresponds to Bernstein's reconstruction formulas reviewed in Section 2, the induction hypothesis is satisfied for these two cases.

We now assume that Algorithm 1 is valid up to s and we prove its validity for $s+1$. Specifically, we have to show that when Algorithm 1 is run with inputs $C^{(s+1)}_0, C^{(s+1)}_1,\ldots,C^{(s+1)}_{3^{s+1}-1}$ it correctly computes

$C = A \times B$. We use for this the following expression of $C = C_0^{(0)}$ in terms of $C_0^{(1)}$, $C_1^{(1)}$ and $C_2^{(1)}$:

$$\begin{aligned} C^{(0)} &= (1+X^{n/2})C_0^{(1)} + X^{n/2}C_1^{(1)} + X^{n/2}(1+X^{n/2})C_2^{(1)} \\ &= (1+X^{n/2}) \underbrace{\left(C_0^{(1)} + X^{n/2}C_2^{(1)}\right)}_{(*)} + X^{n/2}C_1^{(1)}. \end{aligned} \tag{11}$$

Now, by induction hypothesis, if we apply the code in Algorithm 1 for a depth s, it correctly reconstructs

- $C_0^{(1)} = A_0^{(1)} \times B_0^{(1)}$ for $n' = n/2$ and inputs $C_0^{(s+1)}, \ldots, C_{3^s-1}^{(s+1)}$.
- $C_1^{(1)} = A_1^{(1)} \times B_1^{(1)}$ for $n' = n'/2$ and inputs $C_{3^s}^{(s+1)}, \ldots, C_{2\cdot 3^{s+1}-1}^{(s+1)}$.
- $C_2^{(1)} = A_2^{(1)} \times B_2^{(1)}$ for $n' = n'/2$ and inputs $C_{2\cdot 3^s}^{(s+1)}, \ldots, C_{3^{s+1}-1}^{(s+1)}$.

Our strategy is to merge this valid code which computes $C_0^{(1)}$ and $C_2^{(1)}$ and arrange this code in order to obtain a code which computes the term $C_0^{(1)} + X^{n/2}C_2^{(1)}$ in $(*)$ in (11) and afterwards to derive the code which computes $C = C_0^{(0)}$. The merged code computing $C_0^{(1)} + X^{n/2}C_2^{(1)}$ is given in Algorithm 2. We have merged the code by performing one simple change: when an accumulation is performed, we accumulate in the same variable U the values corresponding to $C_0^{(1)}$ and the values corresponding to $C_2^{(1)}$ multiplied by $X^{n/2}$.

Algorithm 2 Two-Merged GBR$_s$

Require: $C_0^{(s+1)}, \ldots, C_{3^s-1}^{(s+1)}$ and $C_{2\cdot 3^s}^{(s+1)}, \ldots, C_{3^{s+1}-1}^{(s+1)}$.
Ensure: $U = \text{GBR}_s(C_0^{(s+1)}, \ldots, C_{3^s-1}^{(s+1)}) + X^{n'}\text{GBR}_{2,s}(C_{2\cdot 3^s}^{(s+1)}, \ldots, C_{3^{s+1}-1}^{(s+1)})$

$U \leftarrow \left(\sum_{i=0}^{2^s-1} C_{\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}}\right) + X^{n'}\left(\sum_{i=0}^{2^s-1} C_{2\cdot 3^s+\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}}\right)$ *//Merged initializations*

for $h = s$ **to** 1

  $U \leftarrow U \times (1 + X^{\frac{n'}{2^{h+1}}})$ *//Merged multiplications of depth h*

  for $i = 0$ **to** $2^{h-1} - 1$ } *// reconstruction // of the L/R-then-M terms*

    $j \leftarrow 3\sigma(i) + 1$

    $V_i \leftarrow \text{GBR}_{s-h}(C_{3^{s-h}j}^{(s+1)}, \ldots, C_{3^{s-h}j+3^{s-h}-1}^{(s+1)})$

    $V_i' \leftarrow \text{GBR}_{s-h}(C_{2\cdot 3^s+3^{s-h}j}^{(s+1)}, \ldots, C_{2\cdot 3^s+3^{s-h}j+3^{s-h}-1}^{(s+1)})$

  end for

  $U \leftarrow U + X^{\frac{n'}{2^h}}\left(\sum_{i=0}^{2^{h-1}-1} V_i X^{\frac{in'}{2^{h-1}}}\right) + X^{n'}X^{\frac{n'}{2^h}}\left(\sum_{i=0}^{2^{h-1}-1} V_i' X^{\frac{in'}{2^{h-1}}}\right)$ *//Merged accumulation of //the L/R-then-M terms*

end for

We now arrange each step of Algorithm 2:

- *Modification of the merged initializations.* Using the definition of σ in Lemma 1, for each $i \in \{0, 1, \ldots, 2^s - 1\}$, we have $2 \cdot 3^s + \sigma(i) = \sigma(2^s + i)$. We can then arrange the merged initializations step as follows

$$\begin{aligned} \left(\sum_{i=0}^{2^s-1} C_{\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}}\right) + X^{n'}\left(\sum_{i=0}^{2^s-1} C_{2\cdot 3^s+\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}}\right) &= \left(\sum_{i=0}^{2^s-1} C_{\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}}\right) + \left(\sum_{i=0}^{2^s-1} C_{\sigma(2^s+i)}^{(s+1)} X^{\frac{(2^s+i)n'}{2^s}}\right) \\ &= \left(\sum_{i=0}^{2^s-1} C_{\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}}\right) + \left(\sum_{i=2^s}^{2^{s+1}-1} C_{\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}}\right) \\ &= \sum_{i=0}^{2^{s+1}-1} C_{\sigma(i)}^{(s+1)} X^{\frac{in'}{2^s}} \end{aligned}$$

- *Modification of the merged reconstruction of the L/R-then-M terms.* We arrange the loop *"**for** $i = 0$ **to** $2^{h-1} - 1$"* by splitting this loop into two loops: a loop on V_i and a loop on V'_i. For the loop on V_i we only change the names of the variables: i becomes i' and j becomes j'. For the second loop on V'_i we perform changes in the variables j and i. Specifically, for i we perform the change of variable $i' = i + 2^{h-1}$. For the variable j we remark that $2 \cdot 3^s + 3^{s-h}j = 3^{s-h}(2 \cdot 3^h + j)$ which implies that $j' = 2 \cdot 3^h + j$ satisfies

$$\begin{aligned} j' &= 2 \cdot 3^h + 3\sigma(i) + 1 \\ &= 3(2 \cdot 3^{h-1} + \sigma(i)) + 1 \\ &= 3\sigma(2^{h-1} + i) + 1 \\ &= 3\sigma(i') + 1. \end{aligned}$$

  We obtain the following modified pseudo-code:

  for $i' = 0$ **to** $2^{h-1} - 1$ **do**
  $j' \leftarrow 3\sigma(i') + 1$
  $V_{i'} \leftarrow \mathrm{GBR}_{s-h}(C^{(s+1)}_{3^{s-h}j'}, \ldots, C^{(s+1)}_{3^{s-h}j'+3^{s-h}-1})$
  end for
  for $i' = 2^{h-1}$ **to** $2^h - 1$ **do**
  $j' \leftarrow 3\sigma(i') + 1$
  $V'_{i'-2^h} \leftarrow \mathrm{GBR}_{s-h}(C^{(s+1)}_{3^{s-h}j'}, \ldots, C^{(s+1)}_{3^{s-h}j'+3^{s-h}-1})$
  end for

  The two **for** loops in the above pseudo-code can then be merged into a single **for** loop. We just have to rename $V'_{i'-2^{h-1}}$ by $V_{i'}$, $i' = 2^{h-1}, \ldots, 2^h - 1$. We obtain the following loop:

  for $i' = 0$ **to** $2^h - 1$ **do**
  $j' \leftarrow 3\sigma(i') + 1$
  $V_{i'} \leftarrow \mathrm{GBR}_{s-h}(C^{(s+1)}_{3^{s-h}j'}, \ldots, C^{(s+1)}_{3^{s-h}j'+3^{s-h}-1})$
  end for

- *Modification of the accumulation of L/R-then-M terms.* Here we first need to replace $V'_i = V'_{i'-2^{h-1}}$ by $V_{i'}$ due to the former changes and then perform the following simplifications

$$\begin{aligned} & U + X^{\frac{n'}{2^h}} \left(\sum_{i'=0}^{2^{h-1}-1} V_{i'} X^{\frac{i'n'}{2^{h-1}}} \right) + X^{n'} X^{\frac{n'}{2^h}} \left(\sum_{i'=2^{h-1}}^{2^h-1} V_{i'} X^{\frac{(i'-2^{h-1})n'}{2^{h-1}}} \right) \\ = \; & U + X^{\frac{n'}{2^h}} \left(\sum_{i'=0}^{2^{h-1}-1} V_{i'} X^{\frac{i'n'}{2^{h-1}}} \right) + X^{\frac{n'}{2^h}} \left(\sum_{i'=2^{h-1}}^{2^h-1} V_{i'} X^{\frac{i'n'}{2^{h-1}}} \right) \\ = \; & U + X^{\frac{n'}{2^h}} \left(\sum_{i'=0}^{2^h-1} V_{i'} X^{\frac{i'n'}{2^{h-1}}} \right). \end{aligned}$$

These modifications, along with the change $h' = h+1$ and $n' = n/2$ and the trivial change $i = i'$ and $j = j'$ results in the following pseudo-code which correctly computes $C^{(1)}_0 + X^{n/2} C^{(1)}_2$.

$U \leftarrow \sum_{i=0}^{2^{s+1}-1} C^{(s+1)}_{\sigma(i)} X^{\frac{in}{2^{s+1}}}$
for $h' = s + 1$ **to** 2 **do**
  $U \leftarrow U \times (1 + X^{\frac{n}{2^{h'}}})$

for $i = 0$ **to** $2^{h'-1} - 1$ **do**

$j \leftarrow 3\sigma(i) + 1$

$V_i \leftarrow \text{GBR}_{s+1-h'}(C^{(s+1)}_{3^{s+1-h'}j}, \ldots, C^{(s+1)}_{3^{s+1-h'}j+3^{s+1-h'}-1})$

end for

$U \leftarrow U + X^{\frac{n}{2^{h'}}} \left(\sum_{i=0}^{2^{h'-1}-1} V_i X^{\frac{in}{2^{h'-1}}} \right)$

end for

We finally obtain the code of Algorithm 1 for $s+1$ by adding the computation $U \leftarrow (1 + X^{n/2})U$ and $U \leftarrow U + \text{GBR}_s(C^{(s+1)}_{3^s}, \ldots, C^{(s+1)}_{2\cdot 3^s-1})$. But these two operations correspond to $h' = 1$ in the previous pseudo code and the resulting pseudo-code correctly computes $C = C^{(0)}_0$ as in (11). This concludes the proof on the validity of Algorithm 1. □

3.4 Complexity evaluation

We evaluate in this subsection the complexity of Algorithm 1 and of the resulting optimized Karatsuba multiplier. We first evaluate the arithmetic complexity of Algorithm 1, i.e., the number of bit additions required in the reconstruction.

3.4.1 Arithmetic complexity of the GBR$_s$ algorithm

The number of bit additions performed in Algorithm 1 is denoted as $\mathcal{S}(n, s)$. We evaluate this complexity in two steps: we first provide in Lemma 2 a recursive form of the arithmetic complexity of Algorithm 1. We will then derive a regular non-recursive form of this complexity in a second lemma.

Lemma 2 (Recursive complexity of the GBR$_{2,s}$ algorithm)**.** *The number of bit additions $\mathcal{S}(s, n)$ performed in the generalized Bernstein's reconstruction of depth s (Algorithm 1) is as follows*

$$\mathcal{S}(s, n) = n + s(2n - 1) - \frac{n}{2^s} - 2^{s+1} + 2 + \sum_{i=1}^{s} 2^{i-1} \mathcal{S}(s - i, n/2^i) \tag{12}$$

Proof: We separately evaluate the cost of the four steps of Algorithm 1.

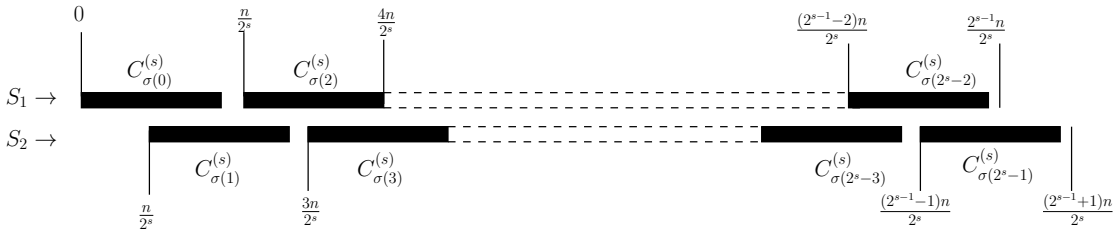
- *Cost of Step 1, the initialization step.* We evaluate here the cost of the initialization step $U \leftarrow (\sum_{i=0}^{2^s-1} C^{(s)}_{\sigma(i)} X^{\frac{in}{2^s}})$. We arrange the sum as follows

$$\left(\sum_{i=0}^{2^s-1} C^{(s)}_{\sigma(i)} X^{\frac{in}{2^s}} \right) = \underbrace{\left(\sum_{i=0}^{2^{s-1}-1} C^{(s)}_{\sigma(2i)} X^{\frac{2in}{2^s}} \right)}_{S_1} + \underbrace{\left(\sum_{i=0}^{2^{s-1}-1} C^{(s)}_{\sigma(2i+1)} X^{\frac{(2i+1)n}{2^s}} \right)}_{S_2}.$$

  As we can see in Fig 7, there are no overlaps in the sums S_1 and S_2. The computations consists of adding the 2^{s-1} terms $C^{(s)}_{\sigma(1)}, \ldots, C^{(s)}_{\sigma(2^{s-1}-1)}$ of S_2 to S_1. Since each $C^{(s)}_{\sigma(j)}$ has size $2n/2^s - 1$, the addition of the $2^{s-1} - 1$ terms $C^{(s)}_{\sigma(1)}, \ldots, C^{(s)}_{\sigma(2^{s-1}-3)}$ requires $\frac{2n}{2^s} - 2$ bit additions, and the last term $C^{(s)}_{\sigma(2^{s-1}-1)}$ requires $\frac{n}{2^s} - 1$ bit additions. This leads to the following cost of this step:

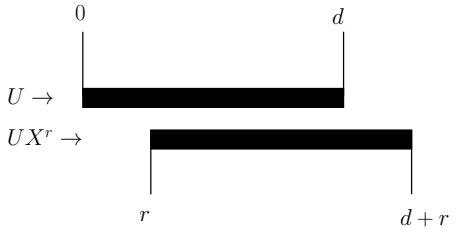
$$(2^{s-1} - 1)(\tfrac{2n}{2^s} - 2) + \tfrac{n}{2^s} - 1 \quad = \quad n - n/2^s - 2^s + 1.$$

Fig. 7. Evaluation of the cost of the initialization step



We can also remark in Fig. 7 that the degree of U after this first step is $d_I = (2^s+1)\frac{n}{2^s} - 2$.

- *Cost of Step 2.* In this step we sequentially multiply U by $1 + X^{n/2^h}$ for $h = s, s-1, \ldots, 1$. Let us denote d_I as the initial degree of U. The following diagram shows that the cost of a multiplication of a polynomial U of degree d by $1 + X^r$ requires $d+1-r$ bit additions and results in a polynomial of degree $d + r$.



We can then obtain the costs of each multiplication by $1 + X^{n/2^h}$ for $h = s, s-1, \ldots, 1$ and also the degree of the variable U after each multiplication. These costs and degrees are given in Table 2. We obtain the total cost of Step 2 after noticing that each multiplication in Table 2 has a cost of

TABLE 2

Cost evaluation of Step 2

Level	Cost	Resulting degree
s	$d_I + 1 - n/2^s$	$d_I + n/2^s$
$s-1$	$(d_I + 1 + n/2^s) - n/2^{s-1}$	$d_I + n/2^s + n/2^{s-1}$
$s-2$	$(d_I + 1 + n/2^s + n/2^{s-1}) - n/2^{s-2}$	$d_I + n/2^s + n/2^{s-1} + n/2^{s-2}$
$\vdots$	$\vdots$	$\vdots$
1	$(d_I + 1 + n/2^s + n/2^{s-1} + \cdots + n/4) - n/2$	$d_I + n/2^s + n/2^{s-1} + \cdots + n/4 + n/2 = 2d_I(1 - \frac{1}{2^s})$
Total	$s(d_I + 1 - n/2)$	

$d_I + 1 - n/2^s$. This implies that the cost of Step 2 is

$$\begin{aligned} s(d_I + 1 - n/2) &= s((2^s+1)\tfrac{n}{2^s} - 1 - n/2^s) \\ &= s(n + n/2^s - 1 - n/2^s) \\ &= s(n-1). \end{aligned}$$

Remark 3. The cost of Step 2 can be independently evaluated from Step 3 and Step 4. Indeed, Step 3 and Step 4 do not change the degree of U: the added polynomial in Step 4 for a depth h has a degree of $n/2^h + (2^{h-1}-1)n/2^{h-1} + 2n/2^h - 2 = n + n/2^h - 2$ which is lower than the degree of U since

$$\begin{aligned} \deg U &= d_I + n/2^s + n/2^{s-1} + n/2^{s-2} + \cdots + n/2^h \\ &= (2^s+1)\tfrac{n}{2^s} + n/2^s + n/2^{s-1} + n/2^{s-2} + \cdots + n/2^h \\ &= n - 2 + n/2^s + n/2^s + n/2^{s-1} + n/2^{s-2} + \cdots + n/2^h \\ &= n - 2 + n/2^{h-1} \end{aligned}$$

- *Cost of Step 3.* For a fixed h we have to reconstruct the 2^{h-1} middle terms $C^{(h)}_{3\sigma(i)+1}$, $i = 1, \ldots, 2^{h-1}-1$ using a recursive call to the GBR$_{s-h}$ algorithm. Since, by induction, the GBR$_{s-h}$ algorithm requires $\mathcal{S}(s-h, n/2^h)$, this results in $2^{h-1}\mathcal{S}(s-h, n/2^h)$ bit additions for a given h. Now since h takes all the values among $\{s, s-1, \ldots, 2, 1\}$ and since a reconstruction of depth 0 is free of computation, we conclude that the cost of the recursive reconstructions in the whole algorithm is as follows

$$\sum_{h=1}^{s-1} 2^{h-1}\mathcal{S}(s-h, n/2^h) = \sum_{h'=1}^{s-1} 2^{s-h'-1}\mathcal{S}(h', n/2^{s-h'}).$$

  after performing the change $h' = s - h$ in the summation variable.
- *Cost of Step 4.* For each $h \in \{s, s-1, \ldots, 2, 1\}$ we add to U the 2^{h-1} middle terms $C^{(h)}_{3\sigma(i)+1}$, for $i = 1, 2, \ldots, 2^{h-1}-1$. Since each $C^{(h)}_{3\sigma(i)+1}$ is of degree $2n/2^h - 2$, the cost of Step 4 is thus as follows

$$\sum_{h=1}^{s} 2^{h-1}(2n/2^h - 1) = sn - \sum_{h=1}^{s} 2^{h-1} = sn - 2^s + 1.$$

We finally obtain the following recursive expression of the complexity by adding the cost of the four steps:

$$\begin{aligned} \mathcal{S}(n,s) &= \overbrace{n - n/2^s - 2^s + 1}^{\text{Step 1}} + \overbrace{s(n-1)}^{\text{Step 2}} + \overbrace{\sum_{i=1}^{s-1} 2^{i-1}\mathcal{S}(i, n/2^i)}^{\text{Step 3}} + \overbrace{sn - 2^s + 1}^{\text{Step 4}} \\ &= n + s(2n-1) - \tfrac{n}{2^s} - 2^{s+1} + 2 + \textstyle\sum_{i=1}^{s-1} 2^{i-1}\mathcal{S}(i, n/2^i). \end{aligned} \tag{13}$$

□

Lemma 3. *The non-recursive form of the number of bit additions $\mathcal{S}(s,n)$ of the GBR_s algorithm is as follows*

$$\mathcal{S}(s,n) = \frac{27n}{4}(3/2)^{s-1} - \frac{15}{4}3^{s-1} - 4n - \frac{n}{2^{s+1}} + \frac{5}{4} - \frac{s}{2}. \tag{14}$$

Proof: We proceed to the proof of (14) using an induction proof. Specifically, we denote the term

$$\gamma(s,n) = n + s(2n-1) - \frac{n}{2^s} - 2^{s+1} + 2$$

which appears in (13) and

$$\rho(s,n) = \frac{27n}{4}(3/2)^{s-1} - \frac{15}{4}3^{s-1} - 4n - \frac{n}{2^{s+1}} + \frac{5}{4} - \frac{s}{2} \tag{15}$$

the expression in the right side of (14). Proving that the two expressions of $\mathcal{S}(n,s)$ of (14) and (13) are equals is equivalent to prove that

$$\sum_{i=1}^{s-1} 2^{i-1}\mathcal{S}(s-i, n/2^i) = \rho(s,n) - \gamma(s,n). \tag{16}$$

We prove that this latter equality is true it by induction on s. For $s=1$ this can be checked directly using the material of Section 2. We assume that (14) and (16) are true up to $s-1$ and we show that they are also true for s. We begin by rewriting the sum in the left side of (16) as follows

$$\begin{array}{rcl} \sum_{i=1}^{s-1} 2^{i-1}\mathcal{S}(s-i,n/2^i) & = & \left(\sum_{i'=1}^{s-2} 2^{s-i'-1}\mathcal{S}(i', n/2^{s-i'})\right) + \mathcal{S}(s-1,n/2) \\ & = & 2\left(\sum_{i'=1}^{s'-1} 2^{s'-i'-1}\mathcal{S}(i', n'/2^{s'-i'})\right) + \mathcal{S}(s',n') \end{array}$$

This latter identity is obtained by setting $s' = s-1$ and $n' = n/2$. We apply the induction hypothesis (16) to the sum $\sum_{i'=1}^{s'-1} 2^{s'-i'-1}\mathcal{S}(i', n'/2^{s'-i'})$ and we use (14) to rewrite $\mathcal{S}(s',n') = \rho(s',n')$

$$\begin{array}{rcl} \sum_{i=1}^{s-1} 2^{i-1}\mathcal{S}(s-i,n/2^i) & = & 2\left(\rho(s',n') - \gamma(s',n')\right) + \rho(s',n') \\ & = & 3\rho(s-1,n/2) - 2\gamma(s-1,n/2). \end{array} \tag{17}$$

Now, we separately arrange the terms $3\rho(s-1,n/2)$ and $2\gamma(s-1,n/2)$. We replace $\rho(s-1,n/2)$ by the corresponding expression in terms of n and s given in (15):

$$\begin{array}{rcl} 3\rho(s-1,n/2) & = & 3\left(\frac{27n}{8}(3/2)^{s-2} - \frac{15}{4}3^{s-2} - 4n/2 - \frac{n}{2^{s+1}} + \frac{5}{4} - \frac{s-1}{2}\right) \\ & = & \frac{27n}{4}(3/2)^{s-1} - \frac{15}{4}3^{s-1} - 6n - \frac{3n}{2^{s+1}} + \frac{15}{4} - \frac{3(s-1)}{2} \\ & = & \left(\frac{27n}{4}(3/2)^{s-1} - \frac{15}{4}3^{s-1} - 4n - \frac{n}{2^{s+1}} + \frac{5}{4} - \frac{s}{2}\right) - 2n - \frac{2n}{2^{s+1}} + \frac{10}{4} + \frac{s}{2} - \frac{3(s-1)}{2} \\ & = & \rho(s,n) - 2n - \frac{n}{2^s} + \frac{10}{4} - \frac{2s}{2} + \frac{3}{2}. \end{array}$$

Similarly we treat the term $2\gamma(s-1,n/2)$ in the same way

$$\begin{array}{rcl} 2\gamma(s-1,n/2) & = & 2\left(n/2 + (s-1)(2n/2-1) - \frac{n}{2^s} - 2^s + 2\right) \\ & = & (n + (s-1)(2n-2) - \frac{n}{2^{s-1}} - 2^{s+1} + 4) \\ & = & (n + s(2n-1) - \frac{n}{2^s} - 2^{s+1} + 2) - (2n-1) - (s-1) - \frac{n}{2^s} + 2 \\ & = & \gamma(s,n) - 2n - s - \frac{n}{2^s} + 4. \end{array}$$

We finally replace in (17) the new expression of $3\rho(s-1,n/2)$ and $2\gamma(s-1,n/2)$. We obtain, after some simplifications, that $\sum_{i=1}^{s-1} 2^{i-1}S(s-i,n/2^i) = \rho(s,n) - \gamma(s,n)$ and this ends the proof. □

3.4.2 Complexities of the parallel multiplier based on GBR$_s$

We consider a parallel multiplier which consists of:

- Two parallel circuits preforming the recursive CPF of depth s for the two inputs A and B.
- 3^s parallel multipliers of polynomial of degree $n/2^s$ which compute $C_i^{(s)} = A_i^{(s)} \times B_i^{(s)}$ for $i = 0, \ldots, 3^s - 1$.
- A reconstruction based on Algorithm 1: the bit additions performed in Step 1, 2 and 4 are performed with parallel XOR gates and the recursive reconstructions in Step 3 are all performed in parallel.

The space and time complexities of this multiplier is established in Lemma 4.

Lemma 4. *The space complexity consists of the number of XOR gates, denoted as $\mathcal{S}_\oplus(n)$, and the number of AND gates, denoted as $\mathcal{S}_\otimes(n)$. The space complexity and the delay of the considered parallel multiplier are as follows*

$$\begin{aligned} \mathcal{S}_{2,\oplus}(n) &= \tfrac{39n}{4}(3/2)^{s-1} - \tfrac{15}{4}3^{s-1} - 6n - \tfrac{n}{2^{s+1}} + \tfrac{5}{4} - \tfrac{s}{2} + 3^s\mathcal{S}_\oplus(n/2^s), \\ \mathcal{S}_{2,\otimes}(n) &= 3^s\mathcal{S}_\otimes(n/2^s), \\ \mathcal{D}(n) &= (2s+1)D_\oplus + \mathcal{D}(n/2^{s+1}). \end{aligned} \tag{18}$$

*Proof: Evaluation of the space complexity.*The number of XOR gates is the sum of the number of XOR gates of the two parallel recursive CPFs of depth s and the 3^s multipliers of degree $n/2^s$ polynomials and the reconstruction of depth s. The number of AND gates is simply equal to $3^s\mathcal{S}_\otimes(n/2^s)$. Based on Fig. 4, we note that the number of XOR gates $\mathcal{S}_{CPF,\oplus}(n)$ of a CPF of depth s satisfies

$$\begin{aligned} \mathcal{S}_{CPF,\oplus}(n) &= n/2 + 3n/2^2 + 9n/2^3 + \cdots + 3^{s-1}n/2^s \\ &= \tfrac{n}{2}\sum_{i=0}^{s-1}\left(\tfrac{3}{2}\right)^i \\ &= \tfrac{n}{2}\tfrac{(\frac{3}{2})^s-1}{\frac{3}{2}-1} \\ &= \left(\left(\tfrac{3}{2}\right)^s - 1\right)n. \end{aligned}$$

We then use the number of XOR gates of the reconstruction of depth s given in (14) to derive the following recursive expression of the space complexity of the proposed multiplier

$$\begin{cases} \mathcal{S}_\oplus(n) &= 2\left(\left(\tfrac{3}{2}\right)^s - 1\right)n + \tfrac{27n}{4}(3/2)^{s-1} - \tfrac{15}{4}3^{s-1} - 4n - \tfrac{n}{2^{s+1}} + \tfrac{5}{4} - \tfrac{s}{2} + 3^s\mathcal{S}_\oplus(n/2^s) \\ \mathcal{S}_\otimes(n) &= 3^s\mathcal{S}_\otimes(n/2^s) \end{cases}$$

After some simplifications we obtain the required expression (18) of the space complexity.

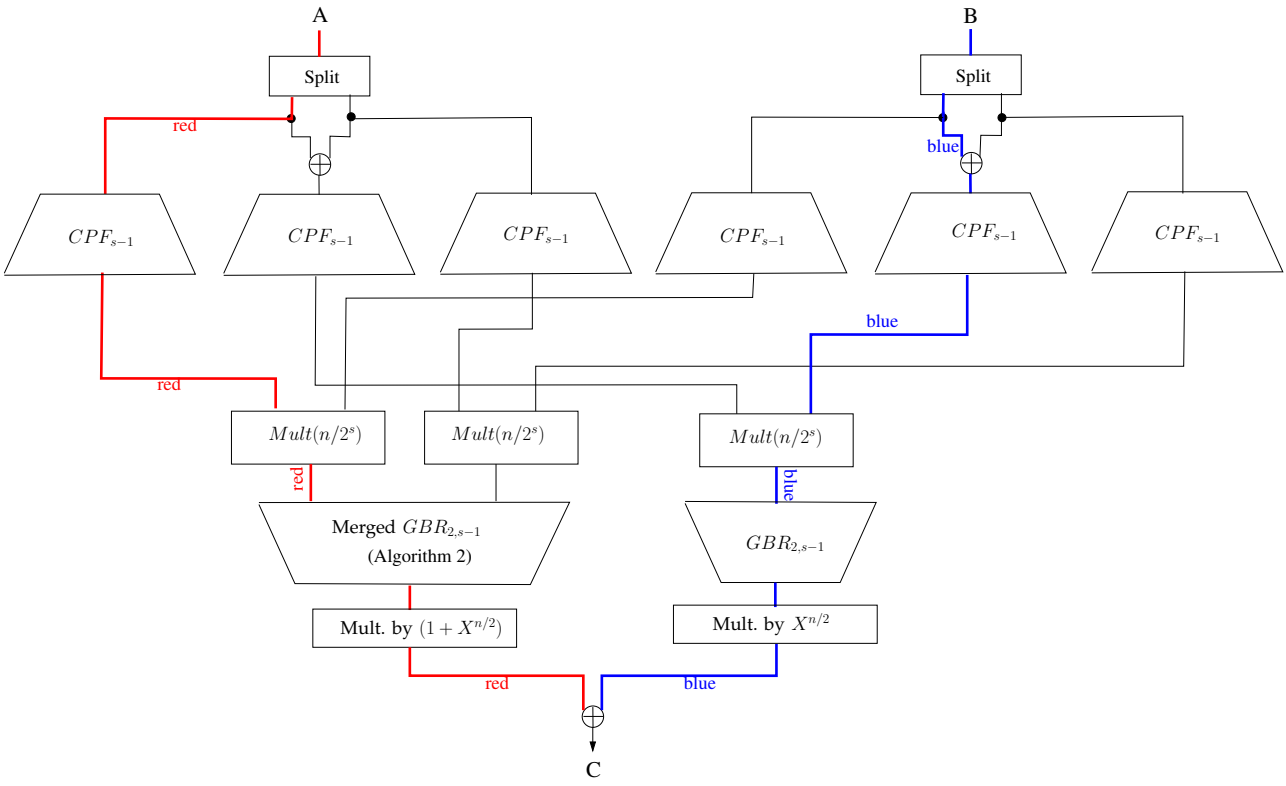
Evaluation of the delay. We show by induction on s that the expression in (18) is true. We assume that the delay in (18) is true up to $s-1$ and we show it for s. For this we decompose the multiplier in the following blocks:

- *Decomposition of the recursive CPF block of depth s.* We re-express the CPF of depth s by applying one recursion of Karatsuba formula and then three blocks of CPF of depth $s-1$. We do this for the two CPF blocks of A and B.
- *Decomposition of the GBR of depth s.* We decompose the reconstruction block of depth s based on the method used in the proof of Theorem 1. We split the reconstruction block in one merged

reconstruction of depth $s-1$ which computes $C_0^{(1)} + X^{n/2}C_2^{(1)}$ and one reconstruction block of depth s for $C_1^{(1)}$. The outputs are connected to a circuit which computes the final operations $(C_0^{(1)} + X^{n/2}C_2^{(1)}) \times (1 + X^{n/2}) + X^{n/2}C_3^{(1)}$.

The resulting decomposition is shown in Fig. 8. The merged GBR reconstruction of depth $s-1$ (Algorithm 2) has the same delay as one single GBR reconstruction of depth $s-1$: we easily see this by comparing Algorithm 2 and Algorithm 1.

Fig. 8. Block decomposition of the multiplier based on merged reconstruction



Consequently, there are two types of critical path:

- The blue path goes through one split block and an addition, followed by one CPF of depth $s-1$, then one multiplier $Mult(n/2^s)$ and then a reconstruction block of depth $s-1$ and a final addition. By induction hypothesis, the path which goes through the CPF block of depth $s-1$ and one multiplier $Mult(n/2^s)$ and then a reconstruction block of depth $s-1$ has a delay of $(2(s-1)+1)D_{\oplus} + \mathcal{D}(n/2^s)$. With the initial and final addition, we obtain the following delay for the entire path

$$\mathcal{D} = (2s+1)D_{\oplus} + \mathcal{D}(n/2^s).$$

- The red path goes through one CPF of depth $s-1$ then one multiplier $Mult(n/2^s)$ and then the merged reconstruction plus the final operations (the multiplication by $1+X^{n/2}$ and the last addition).

By induction hypothesis the delay of one CPF of depth $s-1$ plus one multiplier $Mult(n/2^s)$ plus the merged reconstruction of depth $s-1$ is equal to $(2(s-1)+1)D_\oplus + \mathcal{D}(n/2^s)$. So the critical of this red path has the following delay

$$\mathcal{D} = (2s+1)D_\oplus + \mathcal{D}(n/2^s).$$

□

4 COMPLEXITY COMPARISON AND CONCLUSION

We have presented in the previous section a generalization of Bernstein's optimization of the Karatsuba reconstruction. This optimization can be used to any depth s in the reconstruction. One interesting case is when we apply this approach to the full recursive reconstruction. In the following corollary we give the complexity of the multiplier based on this approach.

Corollary 1. *If we design a parallel Karatsuba multiplier with a generalized Bernstein's reconstruction of depth $s = \log_2(n)$, i.e., the GBR is applied to the full recursive reconstruction, we obtain a multiplier with the following complexity:*

$$\begin{aligned} \mathcal{S}_\oplus(n) &= \tfrac{21}{4} n^{\log_2(3)} - 6n + 3/4 - \tfrac{\log_2(n)}{2}, \\ \mathcal{S}_\otimes(n) &= n^{\log_2(3)}, \\ \mathcal{D}(n) &= (2\log_2(n)+1)D_\oplus + D_\otimes. \end{aligned}$$

The complexity stated in the above corollary is a direct consequence of Lemma 4. In Table 3 we review the complexity of best known approaches to multiply two polynomials of degree $n = 2^k$. This table also contains, for comparison purpose, the complexity of the multiplier specified in Corollary 1 and several other complexities derived from Lemma 4 for several values of s. Note that in these latest cases n is taken as a power of 2^s.

TABLE 3

Space and time complexities of Karatsuba multipliers for $n = 2^k$

Method	# AND	# XOR	Delay
Original Karatsuba [10]	$n^{\log_2(3)}$	$6n^{\log_2(3)} - 8n + 2$	$D_\otimes + 3\log_2(n)D_\oplus$
Fan *et al.* [5]	$n^{\log_2(3)}$	$6n^{\log_2(3)} - 8n + 2$	$D_\otimes + 2\log_2(n)D_\oplus$
Two-way with BR [2]	$n^{\log_2(3)}$	$5.5n^{\log_2(3)} - 7n + 1.5$	$D_\otimes + 3\log_2(n)D_\oplus$
Four-way with BR [2]	$n^{\log_2(3)}$	$5.43n^{\log_2(3)} - 6.8n + 1.375$	$D_\otimes + 2.5\log_2(n)D_\oplus$
Proposed with $s = 3$	$n^{\log_2(3)}$	$5.37n^{\log_2(3)} - 6.68n + 1.30$	$D_\otimes + 2.25\log_2(n)D_\oplus$
Proposed with $s = 4$	$n^{\log_2(3)}$	$5.34n^{\log_2(3)} - 6.61n + 1.27$	$D_\otimes + 2.125\log_2(n)D_\oplus$
Proposed with $s = \log(n)$	$n^{\log_2(3)}$	$5.25n^{\log_2(3)} - 6n + 0.75 - 0.5\log_2(n)$	$D_\otimes + 2\log_2(n)D_\oplus$

The results for $s = 3$ and $s = 4$ were obtained from Lemma 4 and from the fact that the solution of

$f_n = 3^s f_{n/2^s} + cn + d,\ f_0 = 0$ for a fixed s is as follows

$$f_n = \left(\frac{2^s c}{3^s - 2^s} + \frac{d}{3^s - 1}\right) n^{\log_2(3)} - \frac{2^s cn}{3^s - 2^s} - \frac{d}{3^s - 1}.$$

The results presented in Table 3 show that the generalization of Bernstein's approach provides multipliers with better space complexities. This was the initial goal of Bernstein. But surprisingly this also improves the delay of the multiplier. The results in Table 3 even show that the leading term of the complexities (space and time) slowly approach the leading term of the complexity of the multiplier of Corollary 1. This multiplier achieves the best complexity regarding the leading terms of the space and time complexities.

REFERENCES

[1] E.R. Berlekamp. Bit-serial Reed-Solomon encoder. *IEEE Transactions on Information Theory*, IT-28, 1982.

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

[3] D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[4] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *J. Cryptology*, 17(4):297–319, 2004.

[5] H. Fan, J. Sun, M. Gu, and K.-Y. Lam. Overlap-free Karatsuba-Ofman Polynomial Multiplication Algorithm. *IET Information Security*, 4:8–14, March 2010.

[6] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics-Doklady (English translation)*, 7(7):595–596, 1963.

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

[8] E. D. Mastrovito. *VLSI Architectures for Computation in Galois Fields*. PhD thesis, Linkoping University, Department of Electrical Engineering, Linkoping, Sweden, 1991.

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

[10] 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.

[11] 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.