Improved Three-Way Split Approach for Binary Polynomial Multiplication Based on Optimized Reconstruction

Christophe Negre

Improved Three-Way Split Approach for Binary Polynomial Multiplication Based on Optimized Reconstruction

Christophe Negre

Abstract

At Crypto 2009 [1], Bernstein initiated an optimization of Karatsuba formula for binary polynomial multiplication by reorganizing the computations in the reconstruction part of two recursions of the formula. This approach was generalized in [10] to an arbitrary number of recursions resulting in the best known bit parallel multiplier based on Karatsuba formula. In this paper we extend this approach to three-way split formula: we first reorganize two recursions and then extend this re-organization to an arbitrary number s of recursions. We obtain a parallel multiplier with a space complexity of $4.68\, n^{\log_3(6)}+O(n)$ XOR gates and $n^{\log_3(6)}$ AND gates and a delay of $3\log_3(n)D_{\oplus}+D_{\otimes}$. This improves the previous best known results regarding space complexity of [2] and reaches the same time complexity as the the best known approach [4].

Keywords. Binary polynomial multiplication, three-way split formula, optimized recursive reconstruction.

1 INTRODUCTION

Finite field arithmetic are parts of many cryptographic applications like elliptic curve cryptography (ECC) [6], [9], pairing cryptography [5] or even some block-encryption modes like GCM [11] or CWC [7]. Typically, the computations involved in such cryptographic protocols consist in several hundreds or thousands of finite field additions and multiplications. It is thus crucial to have efficient implementation of finite field operations. The two mainly used operations are the addition and the multiplication, but since the addition is quite simple to implement, researchers have focus their effort on efficiently implementing finite field multiplication.

In this paper, we focus on efficient hardware implementation of multiplication in extended binary field, where the field size is supposed to be in the range $[2^{160}, 2^{600}]$ as the ones used in cryptographic protocols. A multiplication in this king of field consists in multiplying two binary polynomials and then reduce the product modulo the irreducible polynomial defining the field. Two approaches are used to perform this modular multiplication: the first one consists to perform separately the multiplication and the reduction,

the second, initiated by Mastrovito [8], performs the multiplication and the reduction at the same time by re-expressing the modular multiplication as a matrix-vector product. This latter approach was improved by Fan and Hasan in [3]: they modified the matrix in order to have Toeplitz matrix, which enabled them to use a subquadratic approach for Toeplitz matrix-vector product. They obtained the best known binary field multiplier in terms of space and time complexities: in [3] the authors report a two-way split multiplier with $5.5n^{\log_2(3)} + O(n)$ XOR gates, $n^{\log_2(3)}$ AND gates and a delay of $2\log_2(n)D_\oplus + D_\otimes$, they also report a three-way split multiplier with $4.8n^{\log_2(3)} + O(n)$ XOR gates, $n^{\log_2(3)}$ AND gates and a delay of $3\log_2(n)D_\oplus + D_\otimes$ where $D_\oplus$ (resp. $D_\otimes$) is the delay of an XOR (resp. AND) gate.

Until recently, the approach which separately performs multiplication and reduction were not competitive compared to the approach based TMVP of [3] since the two-way and three-way subquadratic methods for polynomial multiplication were not competitive compared to their TMVP counterparts. Recently, some progresses have been done on subquadratic polynomial approaches: first, Bernstein in [1] proposed an optimized four-way split approach which reduces the space requirement down to $5.46n^{\log_2(3)} + O(n)$ XOR gates and $n^{\log_2(3)}$ AND gates beating the space requirement of the two-way TMVP multiplier, but the delay remained too large, i.e., $2.5\log_2(n)D_\oplus + D_\otimes$. The approach of Bernstein [1] have been extended in [10], leading to a multiplier with complexity $5.25n^{\log_2(3)} + O(n)$ XOR gates and $n^{\log_2(3)}$ AND gates and a delay $2\log_2(n)D_\oplus + D_\otimes$ providing a better multiplier than the two-way split TMVP multiplier of [3].

Contributions. In this paper we extend the optimization presented in [10] to the case of three-way split formula. We first review the best-known three-way split formula, we then present the proposed optimization in the case of two recursions of the considered formulas. We then generalize the proposed optimization to s recursions: we provide an algorithm and thoroughly prove its validity. We then also provide detailed space and time complexities evaluation. The best three-way split multiplier we obtain has a space complexity of $4.68n^{\log_2(3)} + O(n)$ XOR gates and $n^{\log_2(3)}$ AND gates and a delay of $3\log_3(n)D_\oplus + D_\otimes$.

Organization of the paper. In Section 2, we review the best know three-way split polynomial formula. In Section 3, we present our optimization on two recursions of the considered three-way split formula. In Section 4, we extend this approach to s recursions and establish the complexity results of the resulting multiplier. Finally, in Section 5, we compare the complexity results with the best known approaches and give some concluding remarks.

2 REVIEW OF THREE WAY SPLIT FORMULA

Three-way split formulas for polynomial multiplication are derived from multi-evaluation/interpolation approach or more generally from Chinese remainder theorem. Let us briefly describe how the three-way formulas are obtained. For a more detailed explanation the reader may refer to [12], [13]. Let A and B be two binary polynomials of size $n = 3^\ell$. We split A and B in three parts $A = A_L + A_M X^{n/3} + A_H X^{2n/3}$ and $B = B_0 + B_1 X^{n/3} + B_2 X^{2n/3}$ and then replace $X^{n/3}$ by Y. The two polynomials $A(Y) = A_L + A_M Y + A_H Y$

and $B(Y) = B_L + B_M Y + B_H Y^2$ are considered as degree two polynomials in Y. The product $C(Y) = A(Y) \times B(Y)$ has degree 4, so if it is computed modulo $M(Y) = Y(Y+1)(Y+\infty)(Y^2+Y+1)$ the result remains equal to $A(Y) \times B(Y)$: no reduction is performed since $M(Y)$ has a degree larger than $\deg_Y A(Y) \times B(Y)$. The Chinese remainder theorem can then be used to split up the multiplication $A(Y) \times B(Y)$ modulo $M(Y)$ in four independent modular multiplications: multiplications modulo Y, modulo $Y+1$, modulo $Y+\infty$ and modulo Y^2+Y+1.

$$\begin{aligned}
C(0) &= A(Y)B(Y) \mod Y = A(0)B(0) = A_L B_L, \\
C(1) &= A(Y)B(Y) \mod (Y+1) = A(1)B(1) = (A_L + A_M + A_H)(B_L + B_M + B_H), \\
C(\infty) &= = A(Y)B(Y) \mod (Y+\infty) = A(\infty)B(\infty) = A_H B_H, \\
C(Y) \mod (Y^2+Y+1) &= (A_L + A_M + (A_M + A_H)Y)(B_L + B_H + (B_M + B_H)Y) \mod (Y^2+Y+1),
\end{aligned}$$

In the above operations, the multiplication modulo (Y^2+Y+1) consists in a product of two degree one polynomial in Y and this can be performed with the Karatsuba formula. The Chinese remainder theorem provides also a method to reconstruct C from the four terms $C(0), C(1), C(\infty)$ and $C(Y) \mod (Y^2+Y+1)$. After a number of simplifications and optimizations we can obtain the three-way split formula with six recursive multiplications of [2]:

- *Component polynomial formation (CPF).* The component polynomial formation applied to A consists in splitting A in three parts $A = A_L + A_M X^{n/3} + A_H X^{2n/3}$ with A_L, A_M and A_H of degree $n/3-1$ and then in computing

$$\begin{aligned}
A'_0 &= A_L, & A'_1 &= A_M, & A'_2 &= A_H, \\
A'_3 &= A_L + A_M, & A'_4 &= A_L + A_H, & A'_5 &= A_M + A_H.
\end{aligned}$$

  The above operations require n bit additions. The same is done on B and this results in six polynomials $B'_0, \ldots, B'_5$ of degree $n/3-1$.

- *Recursive products.* This consists in computing the six pairwise products $C'_i, i = 0, 1, \ldots, 5$, as follows

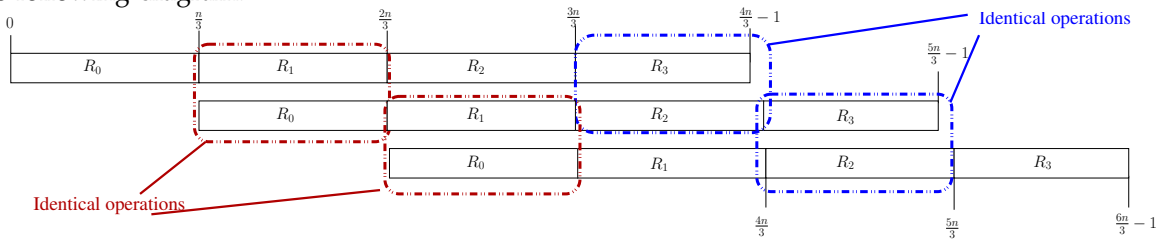
$$\begin{aligned}
C'_0 &= A'_0 B'_0 = A_L B_L, & C'_1 &= A'_1 B'_1 = A_M B_M, \\
C'_2 &= A'_2 B'_2 = A_H B_H, & C'_3 &= A'_3 B'_3 = (A_L + A_M)(B_L + B_M), \\
C'_4 &= A'_4 B'_4 = (A_L + A_H)(B_L + B_H), & C'_5 &= A'_5 B'_5 = (A_M + A_H)(B_M + B_H).
\end{aligned} \tag{1}$$

- *Reconstruction.* The reconstruction of C is performed as:

$$\begin{aligned}
C = \; & C'_0(1 + X^{n/3} + X^{2n/3}) + C'_1 X^{n/3}(1 + X^{n/3} + X^{2n/3}) + C'_2 X^{2n/3}(1 + X^{n/3} + X^{2n/3}) \\
& + C'_3 X^{n/3} + C'_4 X^{2n/3} + C'_5 X^{3n/3} \qquad (2) \\
= \; & \underbrace{(C'_0 + C'_1 X^{n/3} + C'_2 X^{2n/3})}_{R} \times (1 + X^{n/3} + X^{2n/3}) + C'_3 X^{n/3} + C'_4 X^{2n/3} + C'_5 X^{3n/3}. \qquad (3)
\end{aligned}$$

  This is done in three steps as stated in Algorithm 1: the first step initializes R, the second multiplies R by $(1 + X^{n/3} + X^{2n/3})$ and the last step adds the three remaining terms $C'_3 X^{n/3}, C'_4 X^{2n/3}$ and

$C'_5X^{3n/3}$. The only tricky part is the multiplication by $(1+X^{n/3}+X^{2n/3})$. The authors in [2] provide the following diagram:



The identical operations mentioned in the diagram are performed only once during the computation of $R \times (1 + X^{n/3} + X^{2n/3})$, this leads to the reconstruction formula described in Algorithm 1.

Algorithm 1 GBR_1

Require: Six degree $2n/3-2$ polynomials $C'_i, i = 0, \ldots, 5$ defined in (1)

Ensure: R satisfying $R = A \times B$.

// Step 1: Initialization of R

$R \leftarrow C'_0 + C'_1X^{n/3} + C'_2X^{2n/3}$ // costs $2n/3-2$ bit additions

// Step 2: Computation of the product $R \times (1 + X^{n/3} + X^{2n/3})$

$R = R_0 + R_1X^{n/3} + R_2X^{2n/3} + R_3X^{3n/3}$ // Split, no cost

$R'_1 \leftarrow R_0 + R_1$ // costs $n/3$ bit additions

$R'_4 \leftarrow R_2 + R_3$ // costs $n/3-1$ bit additions

$R'_2 \leftarrow R_2 + R'_1$ // costs $n/3$ bit additions

$R'_3 \leftarrow R_1 + R'_4$ // costs $n/3$ bit additions

$R \leftarrow R_0 + R'_1X^{n/3} + R'_2X^{2n/3} + R'_3X^{3n/3} + R'_4X^{4n/3} + R_3X^{5n/3}$ // no cost

// Step 3: Final additions

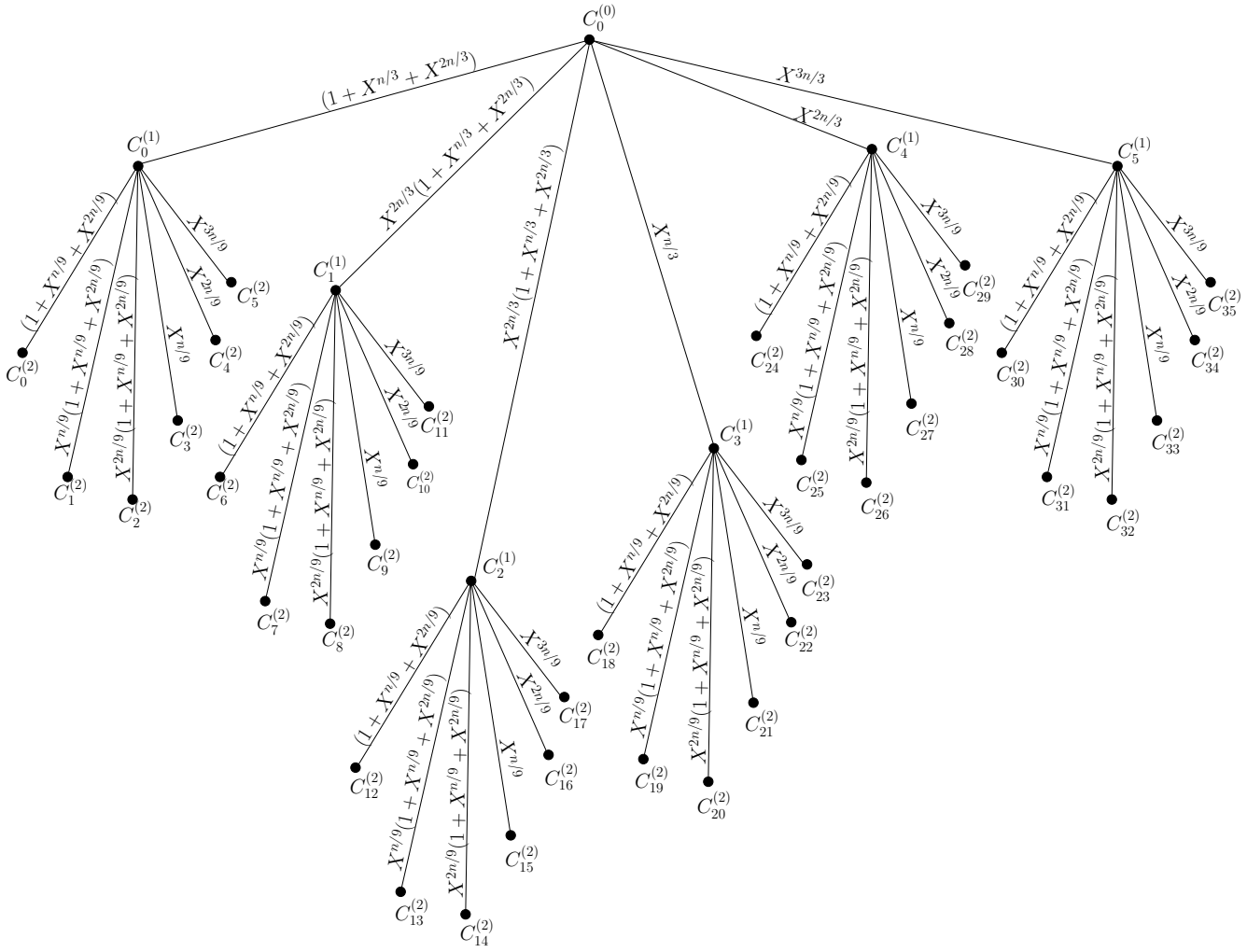
$R \leftarrow R + C'_3X^{n/3} + C'_4X^{2n/3} + C'_5X^{3n/3}$ // costs $3(2n/3-1)$ bit additions

return(R)

We derive the overall space complexity of the three-way split formula expressed in terms of the total number of bit additions $\mathcal{S}_\oplus(n)$ and bit multiplication $\mathcal{S}_\otimes(n)$. We add the cost of the different steps of the formula (component formation and reconstruction) and the cost of six recursive products. The delay is obtained by finding the critical path delay of the formula. This results in the following recursive and non-recursive forms of the complexity:

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

Fig. 1. Reconstruction tree of two recursions of the three-way split formula



3 OPTIMIZATION OF TWO RECURSIONS OF THREE-WAY SPLIT FORMULA

In this section, we present an optimization of the reconstruction of *two recursions* of the three-way split formula presented in Section 2. Two unrolled recursions of the three-way split formula consists in:

1) We first apply two recursions of the component formation to A and B, this produces 36 terms $A_0^{(2)}, A_1^{(2)}, \ldots, A_{35}^{(2)}$, of degree $n/9$ for A and 36 terms $B_0^{(2)}, B_1^{(2)}, \ldots, B_{35}^{(2)}$ for B. These terms are then multiplied two at a time as follows

$$C_i^{(2)} = A_i^{(2)} \times B_i^{(2)} \text{ for } i = 0, 1, \ldots, 35.$$

2) We then perform the reconstruction by applying the three-way split reconstruction formula (3) to each group of six consecutive products $C_{6i+j}^{(2)}, j = 0, \ldots, 5$. We obtain six degree $2n/3-2$ polynomials $C_i^{(1)}$ for $i = 0, 1, \ldots, 5$. Again, we apply the reconstruction formula (3) to $C_0^{(1)}, C_1^{(1)}, \ldots, C_5^{(1)}$ to get

$C = A \times B$.

In the remainder of this section, we focus only on the reconstruction part of the two recursions. We first arrange the two recursions of the reconstruction formula (3) in a reconstruction tree of depth two:

- $C = C_0^{(0)}$ is the root of the tree and it is linked to six children $C_0^{(1)}, C_1^{(1)}, C_2^{(1)}, C_3^{(1)}, C_4^{(1)}$ and $C_5^{(1)}$. Each link is labeled with the factor of $C_i^{(1)}$ which appears in (3). Specifically, the link from $C_0^{(0)}$ to $C_0^{(1)}$ is labeled by $(1+X^{n/3}+X^{2n/3})$, the link from $C_0^{(0)}$ to $C_1^{(1)}$ is labeled by $X^{n/3}(1+X^{n/3}+X^{2n/3})$ and the link from $C_0^{(0)}$ to $C_2^{(1)}$ is labeled by $X^{2n/3}(1+X^{n/3}+X^{2n/3})$. We also label the links from $C_0^{(0)}$ to $C_3^{(1)}, C_4^{(1)}$ and $C_5^{(1)}$ by $X^{n/3}, X^{2n/3}$ and $X^{3n/3}$, respectively.
- Each child $C_i^{(1)}, i = 0, 1, \ldots, 5$ is also linked to six children $C_{6i+j}^{(1)}, j = 0, \ldots, 5$. The links to $C_{6i+j}^{(1)}$ for $j = 0, 1, 2$ are labeled by $X^{jn/9}(1+X^{n/9}+X^{2n/9})$ and the links to $C_{6i+3}^{(i)}, C_{6i+4}^{(i)}$ and $C_{6i+5}^{(i)}$ are labeled by $X^{n/9}, X^{2n/9}$ and $X^{3n/9}$.

The resulting reconstruction tree is depicted in Fig. 1. Now our goal is to modify this reconstruction tree in order to save some computations when performing the multiplications by $(1 + X^{n/3} + X^{2n/3})$ and by $(1 + X^{n/9} + X^{2n/9})$, i.e., by factorizing these multiplications. Following the idea used in [10], we modify the reconstruction tree as follows:

- The term $C_3^{(1)}$ is directly computed from the six inputs $C_{18+j}^{(2)}, j = 0, \ldots, 5$ using the GBR_1 algorithm. We place a block GBR_1 below $C_3^{(1)}$ which represents a reconstruction using Algorithm 1. The inputs of this new block GBR_1 are the six terms $C_{18+j}^{(2)}, j = 0, \ldots, 5$. We do the same modification for $C_4^{(1)}$ and $C_5^{(1)}$.
- We place a block GBR_0 on each link between the leaves $C_i^{(2)}$ for $i \in \{3, 4, 5, 9, 10, 11, 15, 16, 17\}$ and their corresponding parent. The block GBR_0 represents a function which satisfies $\text{GBR}_0(U) = U$ for any U. These modifications are not necessary to derive the optimized reconstruction formula, but it helps to visualize the repetitive pattern in the modified reconstruction tree.
- We finally move down the factor $X^{n/3}$ which appears on the link joining $C_0^{(0)}$ and $C_1^{(1)}$ to the six links joining $C_1^{(1)}$ to his six children. Similarly, we move down the factor $X^{2n/3}$ on the link joining $C_0^{(0)}$ and $C_2^{(1)}$ to the six links from $C_2^{(1)}$ to his six children.

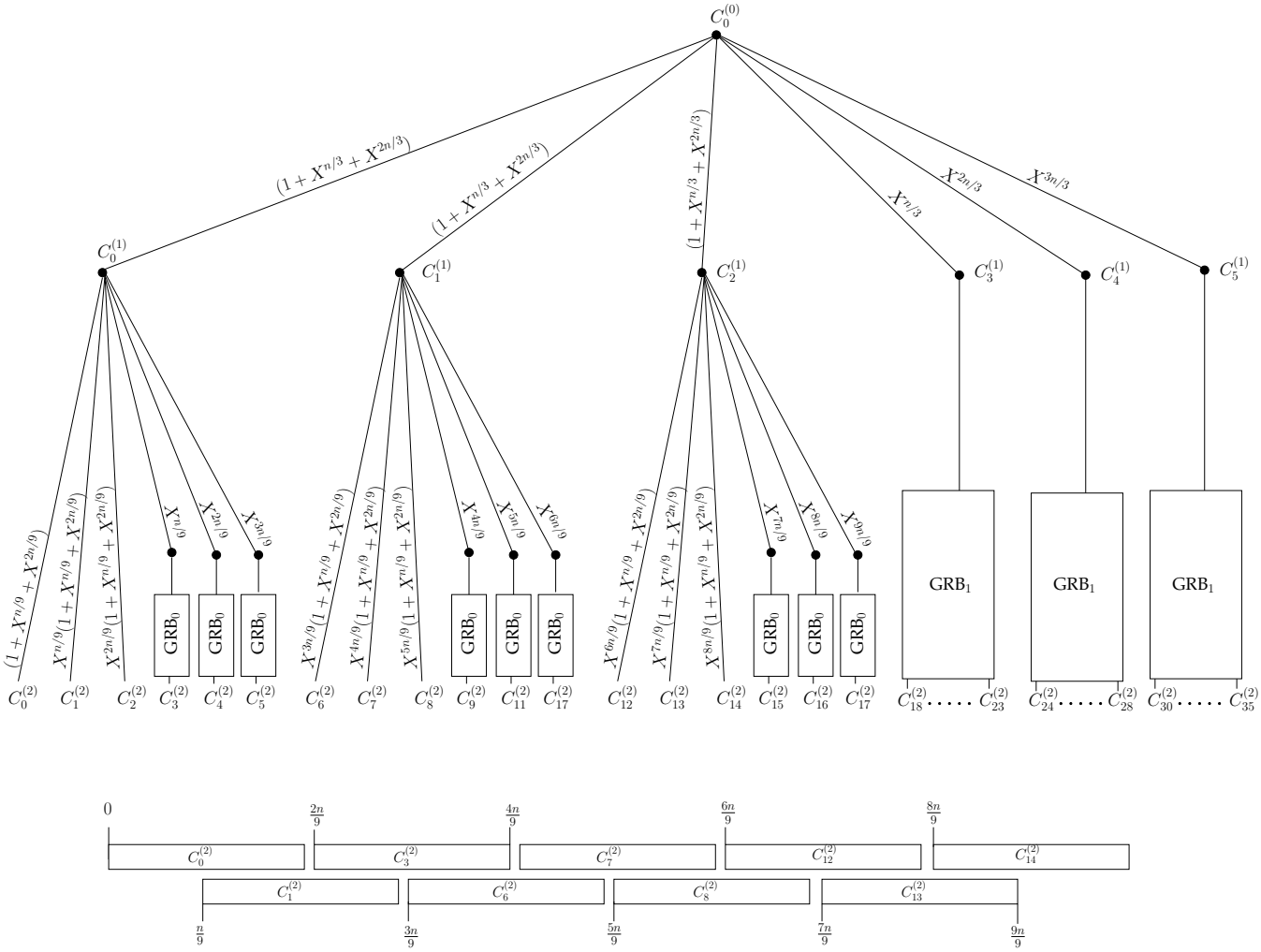
The resulting modified reconstruction tree is shown in Fig. 2. We now derive from this modified reconstruction tree the generalized Bernstein reconstruction of depth 2.

- *Initialization.* We consider the terms $C_i^{(2)}$ such that neither GBR_0, neither GBR_1 are applied to them. We accumulate these terms $C_i^{(2)}$ multiplied by their factor of the form $X^{\alpha_i n/9}$ appearing in the link joining $C_i^{(2)}$ to their parent. This leads to the following expression:

$$\begin{aligned} R_0 \;=\; & C_0^{(2)} + C_1^{(2)} X^{n/9} + C_2^{(2)} X^{2n/9} + C_6^{(2)} X^{3n/9} + C_7^{(2)} X^{4n/9} \\ & + C_8^{(2)} X^{5n/9} + C_{12}^{(2)} X^{6n/3} + C_{13}^{(2)} X^{7n/9} + C_{14}^{(2)} X^{8n/9}. \end{aligned}$$

  The following diagram shows the overlaps involved in the expression of R_0

Fig. 2. Modified reconstruction tree of two recursions of three-way formula

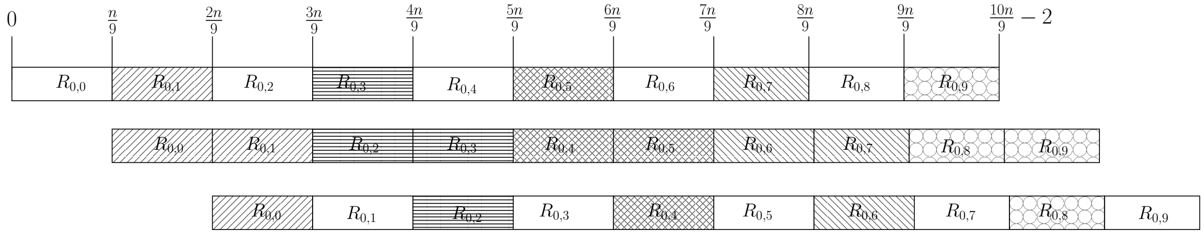


We deduce from this diagram that the cost of this step is equal to $8n/9 - 8$ bit additions and the resulting polynomial R_0 has degree $n + n/9 - 2$.

- *Multiplication of depth 2.* We multiply by the common factor $(1 + X^{n/9} + X^{2n/9})$ appearing in all the label of the accumulated terms $C_i^{(2)}, i \in \{0, 1, 2, 6, 7, 8, 12, 13, 14\}$ in the initialization step:

$$R_1 \quad = \quad R_0(1 + X^{n/9} + X^{2n/9}).$$

We apply the same optimization as in the GBR$_1$ algorithm: we split up R_1 into a number of blocks of size $n/9$ and identify a set of identical additions in $R_0(1 + X^{n/9} + X^{2n/9})$. In the following diagram, we show the splitting of $R_0(1 + X^{n/9} + X^{2n/9})$ along with the identical additions which are filled with the same pattern.



This computation of R_1 necessitates $10n/9-1$ bit additions and R_1 is of degree $n+n/9-2+2n/9 = n+n/3-2$.

- *Accumulation of the reconstructed terms of depth 2.* We add now to R_1 the nine terms $\mathrm{GBR}_0(C_i^{(2)})$ multiplied by their corresponding label

$$\begin{aligned} R_2 \quad = \quad & R_1 + X^{n/9}\Big(\mathrm{GBR}_0(C_3^{(2)}) + \mathrm{GBR}_0(C_4^{(2)})X^{n/9} + \mathrm{GBR}_0(C_5^{(2)})X^{2n/9} \\ & + \mathrm{GBR}_0(C_9^{(2)})X^{3n/9} + \mathrm{GBR}_0(C_{10}^{(2)})X^{4n/9} + \mathrm{GBR}_0(C_{11}^{(2)})X^{5n/9} \\ & + \mathrm{GBR}_0(C_{15}^{(2)})X^{6n/3} + \mathrm{GBR}_0(C_{16}^{(2)})X^{7n/9} + \mathrm{GBR}_0(C_{17}^{(2)})X^{8n/9}\Big). \end{aligned}$$

The nine GBR_0 do not involve any computation. Furthermore, each $\mathrm{GBR}_0(C_i^{(2)})$ has degree $2n/9-2$ and their additions to R_1 contributes to $2n/9-2$ bit additions each. This results in a cost of $2n-9$ bit additions while the degree of R_2 remains equal to $n+n/3-2$.

- *Multiplication of depth 1.* We multiply R_2 by the common factor $(1+X^{n/3}+X^{2n/3})$ which appears in the label of depth 1.

$$R_3 \quad = \quad R_2(1+X^{n/3}+X^{2n/3}).$$

The computation of R_3 is performed as specified in Step 2 of the GBR_1 algorithm, this requires $4n/3-1$ bit additions. The degree of R_3 is equal to $2n-2$.

- *Reconstruction of the three right terms of depth 1.* We use GBR_1 as follows:

$$\begin{aligned} C_3^{(1)} &= \mathrm{GBR}_1(C_{18}^{(2)}, C_{19}^{(2)}, C_{20}^{(2)} C_{21}^{(2)} C_{22}^{(2)}, C_{23}^{(2)}), \\ C_4^{(1)} &= \mathrm{GBR}_1(C_{24}^{(2)}, C_{25}^{(2)}, C_{26}^{(2)} C_{27}^{(2)} C_{28}^{(2)}, C_{29}^{(2)}), \\ C_5^{(1)} &= \mathrm{GBR}_1(C_{30}^{(2)}, C_{31}^{(2)}, C_{32}^{(2)} C_{33}^{(2)} C_{34}^{(2)}, C_{35}^{(2)}). \end{aligned}$$

The complexity evaluation of Section 2 implies that each application of GBR_1 requires $\frac{4n}{3}-6$ bit additions, so this step contributes to $4n-18$ bit additions.

- *Accumulation of the reconstructed terms of depth 1.* We add the reconstructed terms $C_3^{(1)}, C_4^{(1)}$ and $C_5^{(1)}$ multiplied by their corresponding coefficients $X^{n/3}, X^{2n/3}$ and $X^{3n/3}$:

$$C \quad = \quad R_3 + X^{n/3}\left(C_3^{(1)} + C_4^{(1)}X^{n/3} + C_5^{(1)}X^{2n/3}\right).$$

Since each $C_i^{(1)}$ has degree $2n/3-2$ this contributes to $2n-3$ bit additions.

Complexity evaluation. We first evaluate the cost of the component formation. For the polynomial A, the first recursion involves three additions of polynomials of size $n/3$. In the second recursion of the CPF of A we have 6×3 additions of polynomials of size $n/9$. This results in $3n$ bit additions for the CPF of A and the same amount for B.

Algorithm 2 GBR_2

Require: 36 polynomials $C_i^{(2)} = A_i^{(2)} \times B_i^{(2)}, i = 0, \ldots, 35$ of degree $2n/9 - 2$

Ensure: R satisfying $R = A \times B$.

$R \leftarrow C_0^{(2)} + C_1^{(2)} X^{n/9} + C_2^{(2)} X^{2n/9} + C_6^{(2)} X^{3n/9} + C_7^{(2)} X^{4n/9} + C_8^{(2)} X^{5n/9} + C_{12}^{(2)} X^{6n/3} + C_{13}^{(2)} X^{7n/9} + C_{14}^{(2)} X^{8n/9}$

$R \leftarrow R \times (1 + X^{n/9} + X^{2n/9})$

$R \leftarrow R + X^{n/9} \Big(\mathrm{GBR}_0(C_3^{(2)}) + \mathrm{GBR}_0(C_4^{(2)}) X^{n/9} + \mathrm{GBR}_0(C_5^{(2)}) X^{2n/9} + \mathrm{GBR}_0(C_9^{(2)}) X^{3n/9} + \mathrm{GBR}_0(C_{10}^{(2)}) X^{4n/9}$
$+ \mathrm{GBR}_0(C_{11}^{(2)}) X^{5n/9} + \mathrm{GBR}_0(C_{15}^{(2)}) X^{6n/3} + \mathrm{GBR}_0(C_{16}^{(2)}) X^{7n/9} + \mathrm{GBR}_0(C_{17}^{(2)} X^{8n/9}) \Big)$

$R \leftarrow R \times (1 + X^{n/3} + X^{2n/3})$

$C_3^{(1)} \leftarrow \mathrm{GBR}_1(C_{18}^{(2)}, C_{19}^{(2)}, C_{20}^{(2)}, C_{21}^{(2)}, C_{22}^{(2)}, C_{23}^{(2)})$

$C_4^{(1)} \leftarrow \mathrm{GBR}_1(C_{24}^{(2)}, C_{25}^{(2)}, C_{26}^{(2)}, C_{27}^{(2)}, C_{28}^{(2)}, C_{29}^{(2)})$

$C_5^{(1)} \leftarrow \mathrm{GBR}_1(C_{30}^{(2)}, C_{31}^{(2)}, C_{32}^{(2)}, C_{33}^{(2)}, C_{34}^{(2)}, C_{35}^{(2)})$

$R \leftarrow R + X^{n/3} \left(C_3^{(1)} + C_4^{(1)} X^{n/3} + C_5^{(1)} X^{2n/3} \right)$

return(R)

We evaluate now the cost of the reconstruction. We just have to add the contribution of each steps of the proposed GBR_2 algorithm. This is done in Table 1.

TABLE 1

Summary of the costs of each step of GBR_2

Step	Cost
Initialization	$8n/9 - 8$
Multiplication of depth 2	$13n/9 - 1$
Accumulation of depth 2	$18n/9 - 9$
Multiplication of depth 1	$4n/3 - 1$
Three uses of GBR_1 for $C_3^{(1)}, C_4^{(1)}$ and $C_5^{(1)}$	$12n/3 - 18$
Accumulation of depth 1	$6n/3 - 3$
Total	$\frac{35n}{3} - 40$

Finally the overall complexity of the two recursions of three-way split formula is as follows

$$\begin{cases} \mathcal{S}_{\oplus}(n) &= \frac{53n}{3} - 40 + 36\mathcal{S}_{\oplus}(n/9), \\ \mathcal{S}_{\otimes}(n) &= 36\mathcal{S}_{\otimes}(n/9). \end{cases} \tag{4}$$

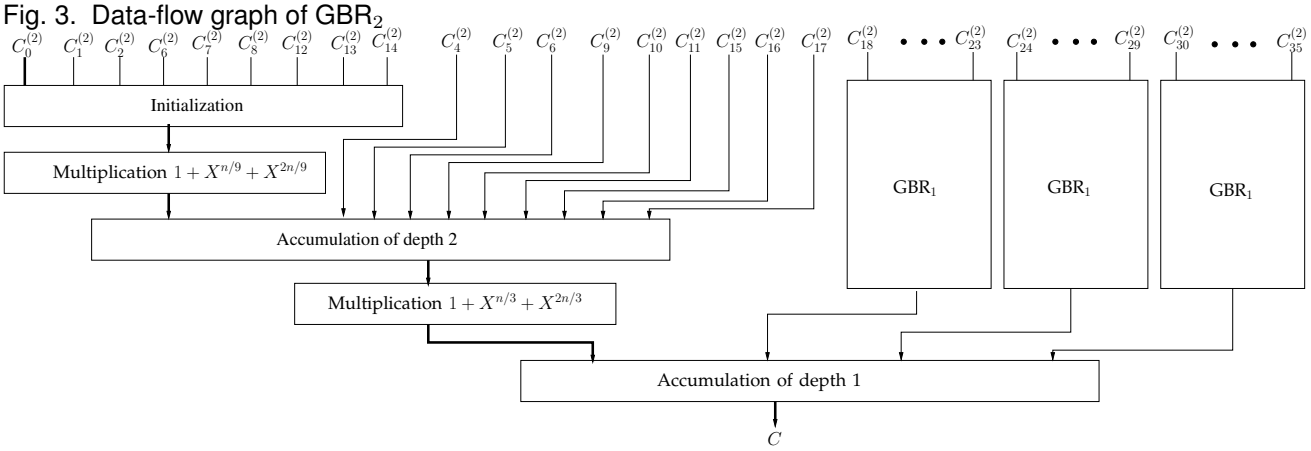
When this method is applied recursively and n is an even power of 3 the above complexity can be re-expressed in the following non-recursive form:

$$\begin{cases} \mathcal{S}_{\oplus}(n) &= \frac{299}{63} n^{\log_3(6)} - \frac{53}{9} n + \frac{8}{7}, \\ \mathcal{S}_{\otimes}(n) &= n^{\log_3(6)}. \end{cases} \tag{5}$$

We now evaluate the delay of the proposed formula. We use the data-flow graph of the proposed reconstruction shown in Fig. 3: the critical path is shown in bold line. Its delay is the sum of $\mathcal{D}(n/9)$ for the

computation of $C_1^{(2)}$ plus $D_\oplus$ for the initialization step, plus $2D_\oplus$ for the multiplication by $1+X^{n/9}+X^{2n/9}$, plus $D_\oplus$ for the accumulation of depth 2, plus $2D_\oplus$ for the multiplication by $1+X^{n/3}+X^{2n/3}$, and $D_\oplus$ for the accumulation of depth 1. This results in a $\mathcal{D}(n) = 7D_\oplus + 9\mathcal{D}(n/9)$ which gives a delay of $\mathcal{D}(n) = \frac{7}{2}\log_3(n)D_\oplus + D_\otimes$ when the recursions are performed entirely.

Fig. 3. Data-flow graph of GBR_2



4 Generalized Bernstein reconstruction of depth s for the three-way formula

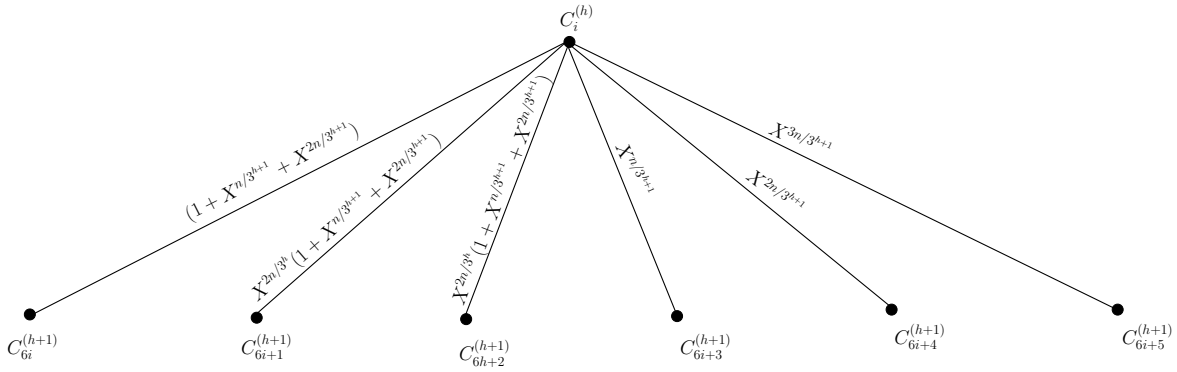
Now, we present the generalization of the optimization obtained on two recursions of the three-way formulas. We first describe the different parts of the three-way split computations when unrolling s recursions: component formations, pairwise products and reconstruction. We then focus on the reconstruction part of the s recursions since it is the part of the computations which can be optimized.

4.1 Recursion of depth s of the three-way split formula

We split the recursion of depth s of the three-way formula into two main steps: a first step which performs the component formations and pairwise products, a second step which performs the reconstruction.

Component formation and pairwise products. The component formation consists in recursively splitting a polynomial in three parts and generate six polynomials with a size divided by three. Specifically, let $A_i^{(h)}$ be one of the terms generated after h recursions and we also assume that $A_i^{(h)}$ is of size $n/3^h$. During the $(h+1)$-th recursion we split $A_i^{(h)}$ in three parts $A_i^{(h)} = A_{iL}^{(h)} + A_{i,M}^{(h)} X^{n/3} + A_{iH}^{(h)} X^{2n/3}$ and then generate the following six polynomials of size $n/3^{h+1}$:

$$\begin{array}{lllllllll} A_{6i}^{(h+1)} & = & A_{i,L}^{(h)}, & A_{6i+1}^{(h+1)} & = & A_{i,M}^{(h)}, & A_{6i+2}^{(h+1)} & = & A_{i,H}^{(h)}, \\ A_{6i+3}^{(h+1)} & = & A_{i,L}^{(h)} + A_{i,M}^{(h)}, & A_{6i+4}^{(h+1)} & = & A_{i,L}^{(h)} + A_{i,H}^{(h)}, & A_{6i+5}^{(h+1)} & = & A_{i,M}^{(h)} + A_{i,H}^{(h)}. \end{array}$$



After s recursions we obtain 6^s terms $A_i^{(s)}$, $i = 0, \ldots, 6^s - 1$ of size $n/3^s$. The same is done for the polynomial B. Then the pairwise products are defined as $C_i^{(s)} = A_i^{(s)} \times B_i^{(s)}, i = 0, \ldots, 6^s - 1$ which provide 6^s polynomials of degree $2n/3^s - 2$.

Reconstruction tree. The reconstruction consists to apply the formula (3) to each group of six consecutive $C_i^{(s)}$ resulting in 6^{s-1} terms $C_i^{(s-1)}$, $i = 0, \ldots, 6^{s-1} - 1$ and then repeat this process recursively until we get the term $C_0^{(0)}$ which is equal to the product $C = A \times B$. This recursive reconstruction of depth s can be arranged in a reconstruction tree of depth s satisfying the following properties:

- The root node is labeled as $C_0^{(0)}$ and corresponds to reconstructed product $C_0^{(0)} = C = A \times B$.
- The intermediate nodes of depth h are labeled as $C_i^{(h)}$ where $0 \leq i < 6^h$. We measure the depth of a node $C_i^{(h)}$ as the length of the upward path joining $C_i^{(h)}$ to the root $C_0^{(0)}$ of the reconstruction tree.
- Each node $C_i^{(h)}$ of depth h is linked down to six children $C_{6i}^{(h+1)}, C_{6i+1}^{(h+1)}, C_{6i+2}^{(h+1)}, C_{6i+3}^{(h+1)}, C_{6i+4}^{(h+1)}$, and $C_{6i+5}^{(h+1)}$ of depth $h+1$. The three left links are labeled by $(1 + X^{n/3^{h+1}} + X^{2n/3^{h+1}}), X^{n/3^{h+1}}(1 + X^{n/3^{h+1}} + X^{2n/3^{h+1}}), X^{2n/3^{h+1}}(1 + X^{n/3^{h+1}} + X^{2n/3^{h+1}})$ and the three right links are labeled with $X^{n/3^{h+1}}, X^{2n/3^{h+1}}$ and $X^{3n/3^{h+1}}$. In the sequel, a link joining $C_i^{(h)}$ to $C_{6i}^{(h+1)}$ or $C_{6i+1}^{(h+1)}$ or $C_{6i+2}^{(h+1)}$ will be called a left (L) link. We will also call any of the links joining $C_{6i}^{(h)}$ to $C_{6i+3}^{(h+1)}$ or $C_{6i+4}^{(h+1)}$ or $C_{6i+5}^{(h+1)}$ a right (R) link.

There are some special nodes in the reconstruction tree which will play important role in the proposed reconstruction algorithm:

Definition 1 (Special nodes)**.** Let $C_i^{(h)}$ be a node of the reconstruction tree.

(i) We say that $C_i^{(h)}$ is an L node if the path joining $C_i^{(h)}$ to the root node $C_0^{(0)}$ comprises only left links. We also say that $C_i^{(h)}$ is an L leaf if $C_i^{(h)}$ is an L node and $h = s$.

(ii) We say that $C_i^{(h)}$ is an L-then-R node if the parent of $C_i^{(h)}$ is an L node and $C_i^{(h)}$ is joined to his parent by a right link.

For example in the reconstruction tree of depth 2 in Fig. 1 we have:

- The nodes $C_3^{(1)}, C_4^{(1)}$ and $C_5^{(1)}$ are L-then-R nodes of depth 1.

- The nodes $C_3^{(2)}, C_4^{(2)}, C_5^{(2)}$ and $C_9^{(2)}, C_{10}^{(2)}, C_{11}^{(2)}$ and $C_{15}^{(2)}, C_{16}^{(2)}, C_{17}^{(2)}$ are L-then-R nodes of depth 2.
- The nodes $C_0^{(1)}, C_1^{(1)}$ and $C_2^{(1)}$ are L nodes of depth 1.
- The nodes $C_0^{(2)}, C_1^{(2)}, C_2^{(2)}$ and $C_6^{(2)}, C_7^{(2)}, C_8^{(2)}$ and $C_{12}^{(2)}, C_{13}^{(2)}, C_{14}^{(2)}$ are L leaves.

The following results concerning the labels of the L and L-then-R nodes of depth h will be useful in sequel. We will use it to list the L nodes or L-then-R nodes of depth h.

Lemma 1. *Let $C_j^{(h)}$ be an L node, then, there exists an integer $i \in [0, 3^h[$ such that $j = \sigma_3(i)$ where $\sigma_3(i) = \sum_{\ell=0}^{h-1} i_\ell 6^\ell$ if $i = (i_{h-1}, \ldots, i_0)_3$ is the representation of i in base 3.*

We leave the proof to the leader (the proof is similar to the proof of Lemma 1 in [10]). Let us just check its validity on few examples:

- The L nodes of depth 1 in the reconstruction tree of depth 2 (Fig. 1) are $C_0^{(1)}, C_1^{(1)}$ and $C_2^{(1)}$. We see that their subscripts $0, 1$ and 2 satisfy $0 = \sigma_3(0)$ and $1 = \sigma_3(1)$ and $2 = \sigma_3(2)$.
- We now consider the L nodes of depth 2. They are grouped into three sets of consecutive subscripts: $C_0^{(2)}, C_1^{(2)}, C_2^{(2)}$ and $C_6^{(2)}, C_7^{(2)}, C_8^{(2)}$ and $C_{12}^{(2)}, C_{13}^{(2)}, C_{14}^{(2)}$. For the first group $C_0^{(2)}, C_1^{(2)}, C_2^{(2)}$ we can check again that $0 = \sigma_3(0)$ and $1 = \sigma_3(1)$ and $2 = \sigma_3(2)$. For the second group $C_6^{(2)}, C_7^{(2)}, C_8^{(1)}$ we have
$$\begin{aligned} 6 &= 0 \times 6^0 + 1 \times 6^1 = \sigma_3(3), \\ 7 &= 1 \times 6^0 + 1 \times 6^1 = \sigma_3(4), \\ 8 &= 2 \times 6^0 + 1 \times 6^1 = \sigma_3(5). \end{aligned}$$
Finally for the last group, the subscripts satisfy:
$$\begin{aligned} 12 &= 0 \times 6^0 + 2 \times 6^1 = \sigma_3(6), \\ 13 &= 1 \times 6^0 + 2 \times 6^1 = \sigma_3(7), \\ 14 &= 2 \times 6^0 + 2 \times 6^1 = \sigma_3(8). \end{aligned}$$
We conclude that all the subscripts of the L leaves are of the form $\sigma_3(i)$ for $i \in \{0, 1, 2, \ldots, 8\} = [0, 3^2[$ as claimed in the lemma.

4.2 The GBR$_s$ algorithm

We now proceed to the generalization of the GBR$_1$ and GBR$_2$ algorithms. In we analyze thoroughly the GBR$_2$ algorithm we can distinguish a repetitive structure:

1) We first start by accumulating the L leaves of the reconstruction tree multiplied by their corresponding factor of the form $X^{in/3^s}$.
2) We then perform the following steps for the depth $h = 2$ and then $h = 1$ of the reconstruction tree:
   a) We multiply R by the factor $(1 + X^{n/3^h} + X^{2n/3^h})$.
   b) We reconstruct all the L-then-R nodes of depth h by applying GBR$_{s-h}$ recursively.
   c) We accumulate all the L-then-R nodes of depth h into R.

We generalize GBR_2 up to s recursions by expanding the above step 2) to $h = s, s-1, s-2, \ldots, 2, 1$. We use for this Lemma 1 to gets the subscripts of the L leaves and the L-then-R nodes for each depth h. This leads us to Algorithm 3.

Algorithm 3 GBR_s

Require: $C_0^{(s)}, \ldots, C_{6^s-1}^{(s)}$
Ensure: $U = \text{GBR}_s(C_0, \ldots, C_{6^s-1})$
$U \leftarrow (\sum_{i=0}^{3^s-1} C_{\sigma_3(i)}^{(s)} X^{\frac{in}{3^s}})$ *//Step 1: Initialization*
for $h = s$ **to** 1
  $U \leftarrow U \times (1 + X^{\frac{n}{3^h}} + X^{\frac{2n}{3^h}})$ *//Step 2: Multiplication of depth h*
  for $i = 0$ **to** $3^h - 1$
    $j \leftarrow \sigma_3(i) + 3$
    $V_i \leftarrow \text{GBR}_{s-h}(C_{6^{s-h}j}^{(s)}, \ldots, C_{6^{s-h}j+6^{s-h}-1}^{(s)})$
  end for *//Step 3 :Reconstruction of the L-then-R terms*
  $U \leftarrow U + X^{\frac{n}{3^h}} \left(\sum_{i=0}^{3^h-1} V_i X^{\frac{in}{3^h}}\right)$ *//Step 4: Accumulation in U of the L-then-R terms*
end for
return(U)

Theorem 1 states the validity of Algorithm 3.

Theorem 1 (Validity of Algorithm 3)**.** *Let us consider two degree n polynomials A and B where $n = 3^s n'$ and let $A_0^{(s)}, \ldots, A_{6^s-1}^{(s)}$ and $B_0^{(s)}, \ldots, B_{6^s-1}^{(s)}$ be their respective component polynomial formation of depth s. Moreover, if $C_i^{(s)} = A_i^{(s)} \times B_i^{(s)}$, $i = 0, \ldots, 6^s - 1$ are their pairwise products, then the application of Algorithm 3 to the inputs $C_0^{(s)}, \ldots, C_{6^s-1}^{(s)}$ returns 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 3 corresponds to the reconstruction of depth 1 and 2 depicted in Algorithms 1 and 2 respectively, the induction hypothesis is satisfied for these two cases.

We now assume that Algorithm 3 is valid up to s and we prove its validity for $s+1$. We have to show that if we run Algorithm 3 with inputs $C_0^{(s+1)}, C_1^{(s+1)}, \ldots, C_{6^{s+1}-1}^{(s+1)}$ where $C_i^{(s+)} = A_i^{(s+1)} \times B_i^{(s+1)}$, $i = 0, \ldots, 6^{s+1} - 1$, it correctly computes $C = A \times B$. By definition we have

$$C^{(0)} = \left(1 + X^{n/3} + X^{2n/3}\right)\left(C_0^{(1)} + X^{n/3} C_1^{(1)} + X^{2n/3} C_2^{(1)}\right) + X^{n/3} C_3^{(1)} + X^{2n/3} C_4^{(1)} + X^{3n/3} C_5^{(1)}. \tag{6}$$

Now, by induction hypothesis, Algorithm 3 correctly reconstructs

- $C_0^{(1)} = A_0^{(1)} \times B_0^{(1)}$ for $n' = n/3$ and inputs $C_0^{(s+1)}, \ldots, C_{6^s-1}^{(s+1)}$,
- $C_1^{(1)} = A_1^{(1)} \times B_1^{(1)}$ for $n' = n/3$ and inputs $C_{6^s}^{(s+1)}, \ldots, C_{2 \cdot 6^s-1}^{(s+1)}$,
- $C_2^{(1)} = A_2^{(1)} \times B_2^{(1)}$ for $n' = n/3$ and inputs $C_{2 \cdot 6^s}^{(s+1)}, \ldots, C_{3 \cdot 6^s-1}^{(s+1)}$.

Our first goal is to merge and arrange the valid code which computes $C_0^{(1)}, C_1^{(1)}$ and $C_2^{(1)}$ in order to obtain a code which computes the term $C_0^{(1)} + X^{n/3}C_1^{(1)} + X^{2n/3}C_2^{(1)}$ in (6). The merged code which computes $C_0^{(1)}, C_1^{(1)}$ and $C_2^{(1)}$ is given in Algorithm 4. This code was obtained by first considering three duplications of the code of Algorithm 3 of depth s with inputs $C_0^{(s+1)}, \ldots, C_{6^s-1}^{(s+1)}$ and $C_{6^s}^{(s+1)}, \ldots, C_{2\cdot 6^s-1}^{(s+1)}$ and $C_{2\cdot 6^s}^{(s+1)}, \ldots, C_{3\cdot 6^s-1}^{(s+1)}$. We then merged these three duplicated codes by accumulating in the same variable U the accumulated terms of the code computing $C_0^{(1)}$ and the accumulated terms of $C_1^{(1)}$ multiplied by $X^{n/3}$ and the accumulated terms multiplied by $X^{2n/3}$ of $C_2^{(1)}$.

Algorithm 4 Merged GBR_s computing $C_0^{(1)} + X^{n/3}C_1^{(1)} + X^{2n/3}C_2^{(1)}$

Require: $C_0^{(s+1)}, \ldots, C_{6^s-1}^{(s+1)}$ and $C_{6^s}^{(s+1)}, \ldots, C_{2\cdot 6^{s+1}-1}^{(s+1)}$ and $C_{2\cdot 6^s}^{(s+1)}, \ldots, C_{3\cdot 6^s-1}^{(s+1)}$ and $n' = n/3$.

Ensure: $U = \text{GBR}_s(C_0^{(s+1)}, \ldots, C_{6^s-1}^{(s+1)}) + X^{n'}\text{GBR}_s(C_{6^s}^{(s+1)}, \ldots, C_{2\cdot 6^s-1}^{(s+1)}) + X^{2n'}\text{GBR}_s(C_{2\cdot 6^s}^{(s+1)}, \ldots, C_{3\cdot 6^s-1}^{(s+1)})$.

$U \leftarrow \left(\sum_{i=0}^{3^s-1} C_{\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right) + X^{n'}\left(\sum_{i=0}^{3^s-1} C_{6^s+\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right) + X^{2n'}\left(\sum_{i=0}^{3^s-1} C_{2\cdot 6^s+\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right)$ *//Merged step 1*

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

$U \leftarrow U \times (1 + X^{\frac{n'}{3^h}} + X^{\frac{2n'}{3^h}})$ *//Merged step 2*

for $i = 0$ **to** $3^h - 1$

$j \leftarrow \sigma_3(i) + 3$

$V_i \leftarrow \text{GBR}_{s-h}(C_{6^{s-h}j}^{(s+1)}, \ldots, C_{6^{s-h}j+6^{s-h}-1}^{(s+1)})$

$V_i' \leftarrow \text{GBR}_{s-h}(C_{6^s+6^{s-h}j}^{(s+1)}, \ldots, C_{6^s+6^{s-h}j+6^{s-h}-1}^{(s+1)})$ *//Merged step 3*

$V_i'' \leftarrow \text{GBR}_{s-h}(C_{2\cdot 6^s+6^{s-h}j}^{(s+1)}, \ldots, C_{2\cdot 6^s+6^{s-h}j+6^{s-h}-1}^{(s+1)})$

end for

$U \leftarrow U + X^{\frac{n'}{3^h}}\left(\sum_{i=0}^{3^h-1} V_i X^{\frac{in'}{3^h}}\right) + X^{n'}X^{\frac{n'}{3^h}}\left(\sum_{i=0}^{3^h-1} V_i' X^{\frac{in'}{3^h}}\right)$ *//Merged step 4*

$+X^{2n'}X^{\frac{n'}{3^h}}\left(\sum_{i=0}^{3^h-1} V_i'' X^{\frac{in'}{3^h}}\right)$

end for

Now we arrange each step of the previous algorithm as follows:

- *Modification of merged step 1.* By definition of σ_3 in Lemma 1, we have $6^s + \sigma_3(i) = \sigma_3(3^s + i)$ and $2 \cdot 6^s + \sigma_3(i) = \sigma_3(2 \cdot 3^s + i)$ for any $i \in \{0, \ldots, 3^s - 1\}$. Then we can arrange the merged step 1 as follows:

$$U = \left(\sum_{i=0}^{3^s-1} C_{\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right) + X^{n'}\left(\sum_{i=0}^{3^s-1} C_{6^s+\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right) + X^{2n'}\left(\sum_{i=0}^{3^s-1} C_{2\cdot 6^s+\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right)$$

$$= \left(\sum_{i=0}^{3^s-1} C_{\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right) + \left(\sum_{i=0}^{3^s-1} C_{\sigma_3(3^s+i)}^{(s+1)} X^{\frac{(3^s+i)n'}{3^s}}\right) + \left(\sum_{i=0}^{3^s-1} C_{\sigma_3(2\cdot 3^s+i)}^{(s+1)} X^{\frac{(2\cdot 3^s+i)n'}{3^s}}\right)$$

where we used that $X^{n'}X^{\frac{in'}{3^s}} = X^{\frac{3^s+in'}{3^s}}$ and $X^{2n'}X^{\frac{in'}{3^s}} = X^{\frac{2\cdot 3^s+in'}{3^s}}$. Now we have

$$U = \left(\sum_{i=0}^{3^s-1} C_{\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right) + \left(\sum_{i=3^s}^{2\cdot 3^s-1} C_{\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right) + \left(\sum_{i=2\cdot 3^s}^{3^{s+1}-1} C_{\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}}\right)$$

$$= \sum_{i=0}^{3^{s+1}-1} C_{\sigma_3(i)}^{(s+1)} X^{\frac{in'}{3^s}},$$

which is the initialization of U in the code of GBR_{s+1}.

- *Modification of the merged step 3 (reconstruction of the L-then-R terms).* We arrange the **for** loop by splitting this loop into three loops: a loop for V_i, a loop for V_i' and a loop for V_i''. We then perform the following changes on the variables i and j of these three loops:
  - In the loop on V_i, we only change the name of the subscripts: i becomes i' and j becomes j'.
  - In the loop on V_i', we first perform the change of variable $i' = i + 3^h$. For the variable j we notice that $6^s + 6^{s-h}j = 6^{s-h}(6^h + j)$ and if we set $j' = 6^h + j$ we have $j' = 6^h + \sigma_3(i) + 3 = \sigma_3(3^h + i) + 3 = \sigma_3(i') + 3$.
  - In the loop on V_i'', we perform the change of variable $i' = i + 2 \cdot 3^h$. For the variable j we notice that $2 \cdot 6^s + 6^{s-h}j = 6^{s-h}(2 \cdot 6^h + j)$ and if we set $j' = 2 \cdot 6^h + j$ it satisfies $j' = 2 \cdot 6^h + \sigma_3(i) + 3 = \sigma_3(2 \cdot 3^h + i) + 3 = \sigma_3(i') + 3$.

  The resulting pseudo-code is depicted below.

  for $i' = 0$ **to** $3^h - 1$ **do**
  $j' \leftarrow \sigma(i') + 3$
  $V_{i'} \leftarrow \mathrm{GBR}_{s-h}(C^{(s+1)}_{6^{s-h}j'}, \ldots, C^{(s+1)}_{6^{s-h}j'+6^{s-h}-1})$
  end for
  for $i' = 3^h$ **to** $2 \cdot 3^h - 1$ **do**
  $j' \leftarrow \sigma(i') + 3$
  $V'_{i'-3^h} \leftarrow \mathrm{GBR}_{s-h}(C^{(s+1)}_{6^{s-h}j'}, \ldots, C^{(s+1)}_{6^{s-h}j'+6^{s-h}-1})$
  end for
  for $i' = 2 \cdot 3^h$ **to** $3^{h+1} - 1$ **do**
  $j' \leftarrow \sigma(i') + 3$
  $V'_{i'-2\cdot 3^h} \leftarrow \mathrm{GBR}_{s-h}(C^{(s+1)}_{6^{s-h}j'}, \ldots, C^{(s+1)}_{6^{s-h}j'+6^{s-h}-1})$
  end for

  The three **for** loops in the former pseudo-code can be merged into a unique **for** loop. We just have to rename $V'_{i'-3^h}$ by $V_{i'}, i' = 3^h, \ldots, 2 \cdot 3^h - 1$ and $V''_{i'-2\cdot 3^h}$ by $V_{i'}, i' = 2 \cdot 3^h, \ldots, 3^{h+1} - 1$. This provides the following loop:

  for $i' = 0$ **to** $3^{h+1} - 1$ **do**
  $j' \leftarrow \sigma(i') + 1$
  $V_{i'} \leftarrow \mathrm{GBR}_{s-h}(C^{(s+1)}_{6^{s-h}j'}, \ldots, C^{(s+1)}_{6^{s-h}j'+6^{s-h}-1})$
  end for

- *Modification of merged step 4 (accumulation of the L-then-R terms).* Here we first need to replace $V_i' = V'_{i'-3^h}$ by $V_{i'}$ for $i' = 3^h, \ldots, 2 \cdot 3^h - 1$ and $V_i'' = V''_{i'-2\cdot 3^h}$ by $V_{i'}$ for $i' = 2 \cdot 3^h, \ldots, 3^{h+1} - 1$ due to the

modifications made in Step 3. We then arrange the accumulation as follows

$$\begin{aligned} & U + X^{\frac{n'}{3^h}} \left(\sum_{i=0}^{3^h-1} V_i X^{\frac{in'}{3^h}} \right) + X^{n'} X^{\frac{n'}{3^h}} \left(\sum_{i=0}^{3^h-1} V_i' X^{\frac{in'}{3^h}} \right) + X^{2n'} X^{\frac{n'}{3^h}} \left(\sum_{i=0}^{3^h-1} V_i'' X^{\frac{in'}{3^h}} \right) \\ = \quad & U + X^{\frac{n'}{3^h}} \left(\sum_{i'=0}^{3^h-1} V_{i'} X^{\frac{i'n'}{3^h}} \right) + X^{n'} X^{\frac{n'}{3^h}} \left(\sum_{i'=3^h}^{2\cdot 3^h-1} V_{i'} X^{\frac{(i'-3^h)n'}{3^h}} \right), \\ & \qquad + X^{2n'} X^{\frac{n'}{3^h}} \left(\sum_{i'=2\cdot 3^h}^{3^{h+1}-1} V_{i'} X^{\frac{(i'-2\cdot 3^h)n'}{3^h}} \right) \end{aligned}$$

the factors $X^{n'}$ and $X^{2n'}$ are then canceled, and we obtain

$$\begin{aligned} = \quad & U + X^{\frac{n'}{3^h}} \left(\sum_{i'=0}^{3^h-1} V_{i'} X^{\frac{i'n'}{3^h}} \right) + X^{\frac{n'}{3^h}} \left(\sum_{i'=3^h}^{2\cdot 3^h-1} V_{i'} X^{\frac{i'n'}{3^h}} \right) + X^{\frac{n'}{3^h}} \left(\sum_{i'=2\cdot 3^h}^{3^{h+1}-1} V_{i'} X^{\frac{i'n'}{3^h}} \right) \\ = \quad & U + X^{\frac{n'}{3^h}} \left(\sum_{i'=0}^{3^{h+1}-1} V_{i'} X^{\frac{i'n'}{3^h}} \right). \end{aligned}$$

These modifications, along with the changes of variables $h' = h+1$ and $n = 3n'$, result in the pseudo-code given in Algorithm 5 which correctly computes $C_0^{(1)} + X^{n/3} C_1^{(1)} + X^{2n/3} C_2^{(1)}$.

Algorithm 5 Modified merged GBR_s

Require: $C_0^{(s+1)}, \ldots, C_{6^s-1}^{(s+1)}$ and $C_{6^s}^{(s+1)}, \ldots, C_{2\cdot 6^{s+1}-1}^{(s+1)}$ and $C_{2\cdot 6^s}^{(s+1)}, \ldots, C_{3\cdot 6^s-1}^{(s+1)}$

Ensure: $U = \mathrm{GBR}_s(C_0^{(s+1)}, \ldots, C_{6^s-1}^{(s+1)}) + X^{n/3}\mathrm{GBR}_s(C_{6^s}^{(s+1)}, \ldots, C_{2\cdot 6^s-1}^{(s+1)}) + X^{2n/3}\mathrm{GBR}_s(C_{2\cdot 6^s}^{(s+1)}, \ldots, C_{3\cdot 6^s-1}^{(s+1)})$

```
U ← Σ_{i=0}^{3^{s+1}-1} C_{σ(i)}^{(s+1)} X^{in/3^{s+1}}
for h' = s+1 to 2 do
    U ← U × (1 + X^{n/3^{h'}} + X^{2n/3^{h'}})
    for i' = 0 to 3^{h'} − 1 do
        j' ← σ_3(i') + 3
        V_{i'} ← GBR_{s+1−h'}(C^{(s+1)}_{6^{s+1−h'} j'}, ..., C^{(s+1)}_{6^{s+1−h'} j' + 6^{s+1−h'} − 1})
    end for
    U ← U + X^{n/3^{h'}} ( Σ_{i'=0}^{3^{h'}−1} V_i' X^{i'n/3^{h'}} )
end for
```

We finally obtain the code of Algorithm 3 for $s+1$ by adding the following two computations corresponding to the multiplication by $(1 + X^{n/3} + X^{2n/3})$ and the additions of the terms $X^{n/3}C_3^{(1)}$, $X^{2n/3}C_4^{(1)}$ and $X^{3n/3}C_5^{(1)}$ in (6):

$$U \leftarrow (1 + X^{n/3} + X^{2n/3})U,$$
$$U \leftarrow U + X^{n/3}\mathrm{GBR}_s(C_{3\cdot 6^s}^{(s+1)}, \ldots, C_{4\cdot 6^s-1}^{(s+1)}) + X^{2n/3}\mathrm{GBR}_s(C_{4\cdot 6^s}^{(s+1)}, \ldots, C_{5\cdot 6^s-1}^{(s+1)}) + X^{3n/3}\mathrm{GBR}_s(C_{5\cdot 6^s}^{(s+1)}, \ldots, C_{6^{s+1}-1}^{(s+1)}).$$

Indeed, these two operations provide the missing loop operations for $h' = 1$ of Algorithm 5. This concludes the proof on the validity of Algorithm 3. □

4.3 Complexity evaluation

This section is dedicated to the complexity evaluation of the GBR_s algorithm and the resulting multiplier based on it.

Lemma 2. *The number of bit additions $\mathcal{S}_{\oplus}(n,s)$ of the GBR$_s$ algorithm satisfies the following recursive expression*

$$\mathcal{S}(s,n) = s\left(\frac{7n}{2}-1\right) - \frac{5}{2}3^s - \frac{n}{4} - \frac{n}{4\cdot 3^{s-1}} + \frac{5}{2} + \sum_{i=1}^{s-1} 3^{s-i}\mathcal{S}(i, n/3^{s-i}). \tag{7}$$

Proof: We evaluate the cost of each step of Algorithm 2 separately. For each step we also provide the resulting degree of U.

- *Cost of Step 1, the initialization step.* We evaluate here the cost of the initialization step

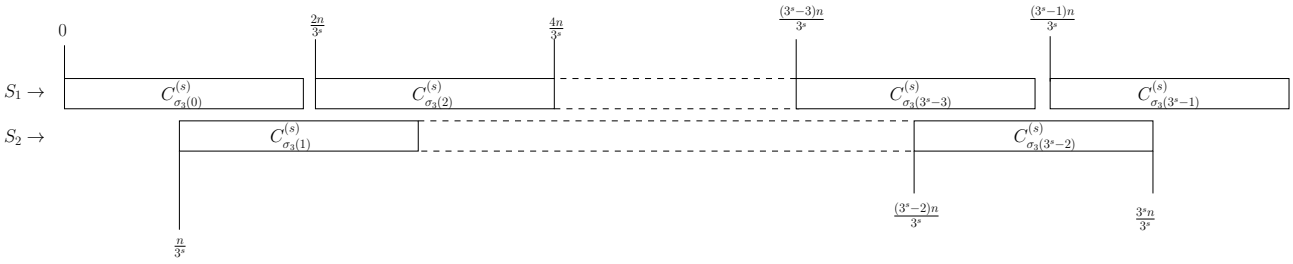
$$U \leftarrow \sum_{i=0}^{3^s-1} C^{(s)}_{\sigma_3(i)} X^{\frac{in}{3^s}}.$$

We arrange the sum as

$$\left(\sum_{i=0}^{3^s-1} C^{(s)}_{\sigma_3(i)} X^{\frac{in}{2^s}}\right) = \underbrace{\left(\sum_{j=0}^{\frac{3^s-1}{2}} C^{(s)}_{\sigma_3(2j)} X^{\frac{2jn}{3^s}}\right)}_{S_1} + \underbrace{\left(\sum_{j=0}^{\frac{3^s-3}{2}} C^{(s)}_{\sigma_3(2j+1)} X^{\frac{(2j+1)n}{3^s}}\right)}_{S_2}.$$

There are no overlaps in S_1 and S_2 and these two sums can be arranged as shown in Fig. 4.

Fig. 4. Cost evaluation of the initialization step

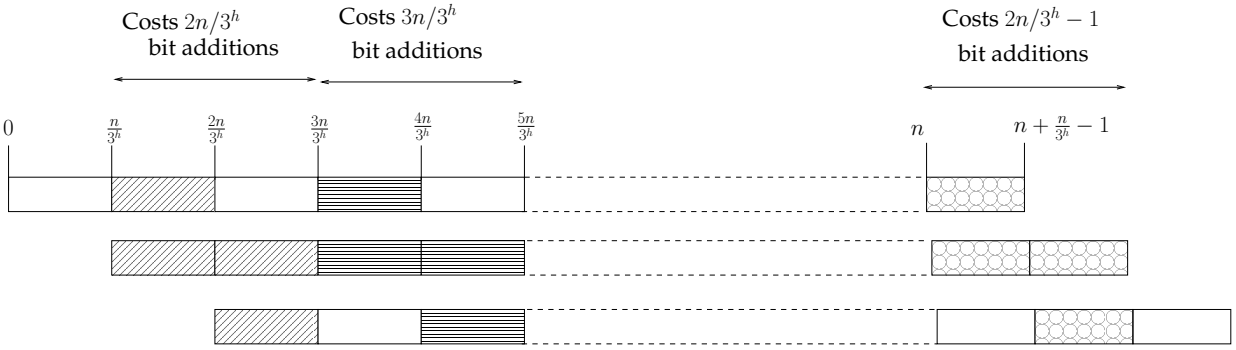


From Fig. 4, we notice that the addition to S_1 of each of the $\frac{3^s-1}{2}$ terms $C^{(s)}_{\sigma_3(1)}, C^{(s)}_{\sigma_3(3)}, \ldots, C^{(s)}_{\sigma_3(3^s-2)}$ of S_2 requires $\frac{2n}{3^s}-2$ bit additions each. Consequently the number of bit additions performed in the initialization step of the GBR$_s$ algorithm is as follows

$$\frac{3^s-1}{2} \times \left(\frac{2n}{3^s}-2\right) = n - \frac{n}{3^s} - 3^s + 1.$$

We also remark from Fig. 4 that the degree of U after this first step is $d_I = (3^s+1)\frac{n}{3^s} - 2 = n + \frac{n}{3^s} - 2$.

- *Cost of Step 2.* In this step we sequentially multiply U by $1 + X^{n/3^h} + X^{2n/3^h}$ for $h = s, s-1, \ldots, 1$. The degree of U starts at $n + \frac{n}{3^s} - 2$ and then changes to $n + \frac{n}{3^{s-1}} - 2$, then $n + \frac{n}{3^{s-2}} - 2$, and so on up to $2n-2$. The computation of the product of U of degree $n + \frac{n}{3^h} - 2$ by $1 + X^{n/3^h} + X^{2n/3^h}$ is performed as follows:



In the above diagram, duplicated additions are filled with the same pattern. The cost of the multiplication by $1 + X^{n/3^h} + X^{2n/3^h}$ is then as follows:

- The left hand side block of size $2n/3^h$ requires $2n/3^h$ bit additions.
- The right hand side block of size $2n/3^h$ and requires $2n/3^h - 1$ bit additions.
- The middle blocks, which again are of size $2n/3^h$, require $3n/3^h$ bit additions each. The number of these blocks is $\frac{(n+n/3^h)-4n/3^h}{2n/3^h} = \frac{3^h-3}{2}$.

Consequently, the cost of a multiplication by $1 + X^{n/3^h} + X^{2n/3^h}$ requires

$$\left(\frac{3^h-3}{2}\right) \times 3n/3^h + 2n/3^h + 2n/3^h - 1 = \frac{3n - n/3^h}{2} - 1 \text{ bit additions.}$$

We report in Table 2 the costs of the multiplications by $1 + X^{n/3^h} + X^{2n/3^h}$ for $h = s, s-1, \ldots, 2, 1$ and the degree of U after these multiplications.

TABLE 2

Cost of the multiplications by $1 + X^{n/3^h} + X^{2n/3^h}$ for $h = s, s-1, \ldots, 1$

Depth	Cost	Resulting degree
$h = s$	$\frac{3n-n/3^s}{2} - 1$	$d_s = d_I + 2n/3^s = n + n/3^{s-1} - 2$
$h = s-1$	$\frac{3n-n/3^{s-1}}{2} - 1$	$d_{s-1} = d_s + 2n/3^{s-1} = n + n/3^{s-2} - 2$
$\vdots$	$\vdots$	$\vdots$
$h = 2$	$\frac{3n-n/3^2}{2} - 1$	$d_2 = d_3 + 2n/3^2 = n + n/3 - 2$
$h = 1$	$\frac{3n-n/3}{2} - 1$	$d_1 = d_2 + 2n/3 = n + n - 2 = 2n - 2$

The sum of the costs of the s multiplications by $(X^{2n/3^h} + X^{n/3^h} + 1), h = s, s-1, \ldots, 1$ given in Table 2 is equal to

$$\begin{aligned} Cost_2 &= s(\tfrac{3n}{2} - 1) - \tfrac{n}{2}(1/3^s + 1/3^{s-1} + \cdots + 1/3) \\ &= s(\tfrac{3n}{2} - 1) - \tfrac{n}{6}\left(\tfrac{1-\frac{1}{3^s}}{1-\frac{1}{3}}\right) \\ &= s(\tfrac{3n}{2} - 1) - \tfrac{n}{4} + \tfrac{n}{4 \cdot 3^s}. \end{aligned}$$

- *Cost of step 3.* The contribution of the recursive computations is as follows:

$$Cost_3 = 3\mathcal{S}(s-1, n/3) + 9\mathcal{S}(s-2, n/9) + \cdots + 3^{s-1}\mathcal{S}(1, n/3^{s-1}) = \sum_{i=1}^{s-1} 3^i \mathcal{S}(s-i, n/3^i).$$

- *Cost of step 4.* For each $h = s, s-1, \ldots, 2, 1$ we add to U the 3^h L-then-R terms $C_j^{(h)}$, $j = 1, \ldots, 3^h - 1$. Each $C_j^{(h)}$ is of degree $2n/3^h - 2$. The cost is thus

$$\begin{aligned} Cost_4 &= 3(2n/3-1) + 9(2n/9-1) + \cdots + 3^s(2n/3^s - 1) \\ &= \textstyle\sum_{i=1}^{s} 3^i(2n/3^i - 1) \\ &= (\textstyle\sum_{i=1}^{s} 2n) - (\sum_{i=1}^{s} 3^i) \\ &= 2sn - \frac{3^{s+1}-3}{2}. \end{aligned}$$

Finally, the total cost of the GBR$_s$ algorithm is as follows:

$$\mathcal{S}(s,n) = \underbrace{n - \frac{n}{3^s} - 3^s + 1}_{Cost_1} + \underbrace{s(\frac{3n}{2} - 1) - \frac{n}{4} + \frac{n}{4 \cdot 3^s}}_{Cost_2} + \underbrace{\sum_{i=1}^{s-1} 3^i \mathcal{S}(s-i, n/3^i)}_{Cost_3} + \underbrace{2sn - \frac{3^{s+1}-3}{2}}_{Cost_4}.$$

This latter expression can be re-written as

$$\mathcal{S}(s,n) = s(\frac{7n}{2} - 1) - \frac{5}{2}3^s + \frac{3n}{4} - \frac{n}{4 \cdot 3^{s-1}} + \frac{5}{2} + \sum_{i=1}^{s-1} 3^{s-i}\mathcal{S}(i, n/3^{s-i}). \tag{8}$$

□

The following lemma provides the non-recursive form of the complexity of the GBR$_s$ algorithm.

Lemma 3. *The solution $\mathcal{S}(s,n)$ of the recursive expression*

$$\sum_{i=1}^{s-1} 3^{s-i}\mathcal{S}(i, n/3^{s-i}) = \mathcal{S}(s,n) - \gamma(s,n) \tag{9}$$

where $\gamma(s,n) = s(\frac{7n}{2} - 1) - \frac{5}{2}3^s + \frac{3n}{4} - \frac{n}{4 \cdot 3^{s-1}} + \frac{5}{2}$ is as follows

$$\mathcal{S}(s,n) = -\frac{7n}{2} + \frac{28}{25} - \frac{28}{25}6^s - \frac{2s}{5} - \frac{n}{10 \cdot 3^{s-1}} + \frac{19n}{5}2^s. \tag{10}$$

Proof: We prove this latter equality by induction on s: we assume that (10) and (9) are true up to $s-1$ and we show that they are also true for s. We begin by rewriting the summation in (9) as follows

$$\begin{aligned} \textstyle\sum_{i=1}^{s-1} 3^{s-i}\mathcal{S}(i, n/3^{s-i}) &= 3\mathcal{S}(s-1, n/3) + \textstyle\sum_{i=1}^{s-2} 3^{s-i}\mathcal{S}(i, n/3^i) \\ &= 3\mathcal{S}(s-1, n/3) + 3\textstyle\sum_{i=1}^{s'-1} 3^{s'-i}\mathcal{S}(i, n'/3^{s'-1}). \end{aligned}$$

The latter identity was obtained by setting $s' = s-1$ and $n' = n/3$. We apply the induction hypothesis to the sum $\sum_{i=1}^{s'-1} 3^{s'-i}\mathcal{S}(i, n'/3^{s'-i})$ and we obtain:

$$\begin{aligned} \textstyle\sum_{i=1}^{s-1} 3^{s-i}\mathcal{S}(i, n/3^{s-i}) &= 3\mathcal{S}(s', n') + 3\left(\mathcal{S}(s', n') - \gamma(s', n')\right) \\ &= 6\mathcal{S}(s-1, n/3) - 3\gamma(s-1, n/3). \end{aligned} \tag{11}$$

Now we separately arrange the terms $6\mathcal{S}(s-1, n/3)$ and $3\gamma(s-1, n/3)$. For $6\mathcal{S}(s-1, n/3)$, we use the induction hypothesis and replace $\mathcal{S}(s-1, n/3)$ by their corresponding expression in term of n and s given

in (10):

$$\begin{aligned} 6\mathcal{S}(s-1,n/3) &= 6\left(-\tfrac{7n}{6}+\tfrac{28}{25}-\tfrac{28}{25}6^{s-1}-\tfrac{2(s-1)}{5}-\tfrac{n}{10\cdot 3^{s-1}}+\tfrac{19n}{15}2^{s-1}\right) \\ &= -7n+6\cdot\tfrac{28}{25}-\tfrac{28}{25}6^s-\tfrac{12s}{5}+6\cdot\tfrac{2}{5}-\tfrac{6n}{10\cdot 3^{s-1}}+\tfrac{19n}{5}2^s \\ &= \left(-\tfrac{7n}{2}+\tfrac{28}{25}-\tfrac{28}{25}6^s-\tfrac{2s}{5}-\tfrac{n}{10\cdot 3^{s-1}}+\tfrac{19n}{5}2^s\right)-\tfrac{7n}{2}+5\cdot\tfrac{28}{25}-\tfrac{10s}{5}+6\cdot\tfrac{2}{5}-\tfrac{5n}{10\cdot 3^{s-1}} \\ &= \mathcal{S}(s,n)-\tfrac{7n}{2}+-2s+8-\tfrac{n}{2\cdot 3^{s-1}}. \end{aligned}$$

We treat the term $3\gamma(s-1,n/2)$ in the same way:

$$\begin{aligned} 3\gamma(s-1,n/3) &= 3\left((s-1)(\tfrac{7n}{6}-1)-\tfrac{5}{2}3^{s-1}+\tfrac{n}{4}-\tfrac{n}{4\cdot 3^{s-1}}+\tfrac{5}{2}\right) \\ &= s(\tfrac{7n}{2}-3)-\tfrac{7n}{2}+3-\tfrac{5}{2}3^s+\tfrac{3n}{4}-\tfrac{3n}{4\cdot 3^{s-1}}+\tfrac{15}{2} \\ &= \left(s(\tfrac{7n}{2}-1)-\tfrac{5}{2}3^s+\tfrac{3n}{4}-\tfrac{n}{4\cdot 3^{s-1}}+\tfrac{5}{2}\right)-2s-\tfrac{7n}{2}+3-\tfrac{2n}{4\cdot 3^{s-1}}+\tfrac{10}{2} \\ &= \gamma(s,n)-2s-\tfrac{7n}{2}-\tfrac{n}{2\cdot 3^{s-1}}+8. \end{aligned}$$

We finally we replace in (11) the new expressions of $6\mathcal{S}(s-1,n/2)$ and $3\gamma(s-1,n/2)$, and make some simplifications

$$\begin{aligned} \textstyle\sum_{i=1}^{s-1}2^{i-1}S(s-i,n/2^i) &= \mathcal{S}(s,n)+\left(-\tfrac{7n}{2}+-2s+8-\tfrac{n}{2\cdot 3^{s-1}}\right)-\gamma(s,n)-\left(-2s-\tfrac{7n}{2}-\tfrac{n}{2\cdot 3^{s-1}}+8\right). \\ &= \mathcal{S}(s,n)-\gamma(s,n). \end{aligned}$$

And this ends the proof. □

Lemma 4. *The number of bit additions $\mathcal{S}_\oplus(n)$ and bit multiplications $\mathcal{S}_\otimes(n)$ involved in the multiplication of two degree $n=3^sn'$ polynomials consisting in parallel component formation of depth s, 6^s parallel multiplications of degree $n/3^s$ polynomials and a GBR$_s$ for the reconstruction are as follows*

$$\left\{\begin{array}{lcl} \mathcal{S}_\oplus(n) &=& -\tfrac{11n}{2}+\tfrac{28}{25}-\tfrac{28}{25}6^s-\tfrac{2s}{5}-\tfrac{n}{10\cdot 3^{s-1}}+\tfrac{29n}{5}2^s+6^s\mathcal{S}_\oplus(n/3^s) \\ \mathcal{S}_\otimes(n) &=& 6^s\mathcal{S}_\otimes(n/3^s) \end{array}\right. \tag{12}$$

The delay of this approach is as follows

$$\mathcal{D}(n)=(3s+1)D_\oplus+\mathcal{D}(n/3^s). \tag{13}$$

Proof: Evaluation of the space complexity. Since, in Lemma 3, we have already established the complexity of the reconstruction, we have only to evaluate the number of bit additions for a component formation of depth s. We prove by induction that a component formation of depth s requires $n(2^s-1)$ bit additions. For $s=1$ we have seen in Section 2 that a CPF requires n bit additions: the induction hypothesis is thus satisfied. Now we assume that the induction hypothesis is true up to s and we prove it for $s+1$. We decompose the $s+1$ recursions of CPF$_{s+1}$ into one recursion which generates six polynomials of degree $n/3$ and then we apply a CPF$_s$ to each of them. The cost of CPF$_{s+1}$ is then: n bit additions for the first recursion plus six times the number of bit additions involved in CPF$_s$ with input of size $n/3$. This results in

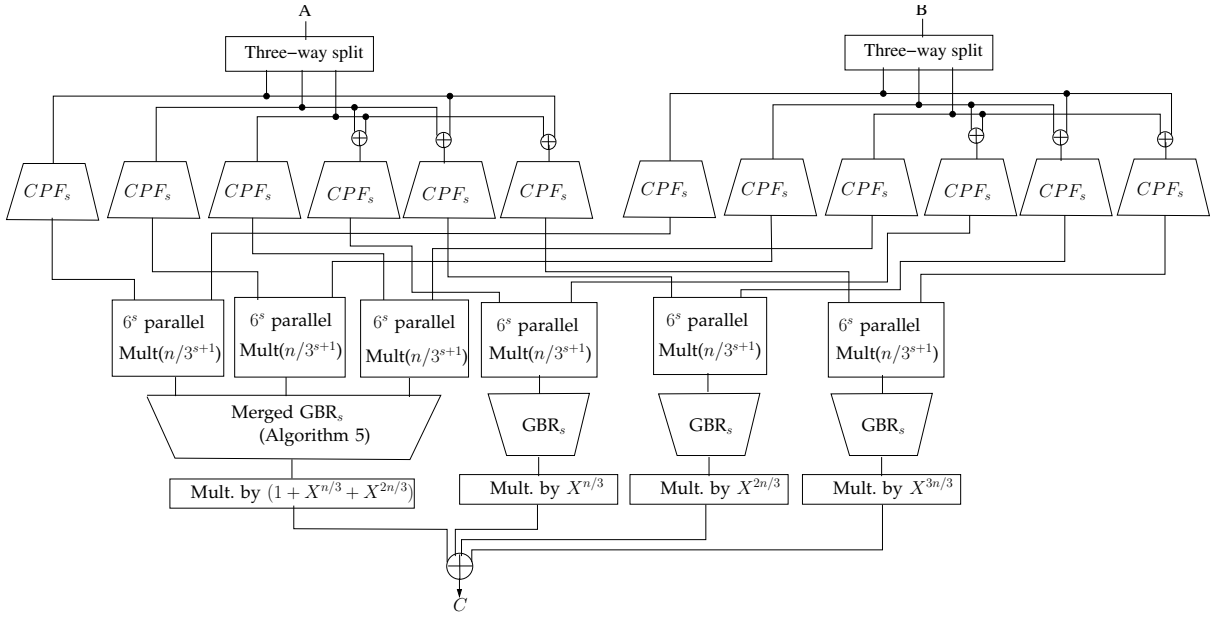
$$n+6\times\frac{n}{3}(2^s-1)=n+n2^{s+1}-2n=n(2^{s+1}-1) \text{ bit additions.}$$

We finally obtain the overall space complexity by adding the complexity of two CPF$_s$ plus 6^s times the complexity of a multiplication of degree $n/3^s$ plus the complexity of GBR$_s$ given in (10). This results in

$$\begin{cases} \mathcal{S}_{\oplus}(n) &= -\frac{11n}{2} + \frac{28}{25} - \frac{28}{25}6^s - \frac{2s}{5} - \frac{n}{10\cdot 3^{s-1}} + \frac{29n}{5}2^s + 6^s\mathcal{S}_{\oplus}(n/3^s), \\ \mathcal{S}_{\otimes}(n) &= 6^s\mathcal{S}_{\otimes}(n/3^s). \end{cases}$$

Evaluation of the delay. Again, we prove it by induction. For $s = 1$ this corresponds to the delay given in Section 2. We now assume that the delay given in the lemma is true for s and we prove it for $s+1$. We have drawn a diagram (Fig. 5) showing the circuit of the considered multiplier. This diagram is based on the proof of validity of Algorithm 3 in which we rewrote GBR$_{s+1}$ in terms of a modified merged GBR$_s$ (Algorithm 5), three GBR$_s$ and a final loop operation. We also know from the proof of Theorem 1 that GBR$_s$ and the merged GBR$_s$ have the same delay since the merging process does not increase the delay. This means that, by induction hypothesis, the circuits based on the GBR$_s$ and merged GBR$_s$ algorithms have a delay of $\mathcal{D}(n/3^s) + (3s+1)D_{\oplus}$.

Fig. 5. Reconstruction of depth $s+1$



The critical path of the circuit of Fig. 5 goes through any of the three left most CPF$_s$ blocks of A or B, then through one $Mul(n/3^{s+1})$ block and then one merged GBR$_s$ block and finally through a multiplication by $1 + X^{n/3} + X^{2n/3}$ and a last accumulation step. By induction hypothesis, the sequence *CPF$_s$ then $Mul(n/3^{s+1})$ and then merged GBR$_s$* contributes to $(3s+1)D_{\oplus} + \mathcal{D}(n/3^{s+1})$ to the critical path delay. The final operations contributes to $3D_{\oplus}$, leading to an overall delay of

$$\mathcal{D}(n) = (3(s+1)+1)D_{\oplus} + \mathcal{D}(n/3^{s+1}).$$

and this concludes the proof. □

In the following corollary, we give the complexity of the multiplier for the special case $s = \log_3(n)$, i.e., if the optimization GBR$_s$ is applied to all the recursions.

Corrolary 1. *Let n be a power of 3, then the three-way split multiplier based on GBR$_s$ when $s = \log_3(n)$ has the following space and time complexities*

$$\begin{cases} \mathcal{S}_\oplus(n) &= \frac{117}{25} n^{\log_3(6)} - \frac{11}{5} n - \frac{2\log_3(n)}{5} + \frac{41}{50}, \\ \mathcal{S}_\otimes(n) &= n^{\log_3(6)}, \\ \mathcal{D}(n) &= 3\log_3(n) D_\oplus + D_\otimes. \end{cases}$$

Proof: We obtain the space complexity directly from Lemma 4 by replacing s by $\log_3(n)$ in (12). For the delay, we notice that the delay of the Initialization steps of GBR$_s$ is null. Indeed, when $s = \log_3(n)$ then the polynomials $C_i^{(s)}$ for $i = 0, 1, \ldots, 6^{\log_3(n)} - 1$ are of degree 0. Consequently, the Initialization step consists in interleaving the coefficients $C^{(s)}_{\sigma_3(i)}, i \in [0, 3^{\log_3(n)}[$ and does not require any delay. So we remove one $D_\oplus$ from the delay of (13) with $s = \log_3(n)$ resulting in the required delay. □

5 Complexity comparison and conclusion

We have presented in this paper a generalization the optimization technique initiated by Bernstein on two recursion of Karatsuba to three-way split formula. We have first presented this technique on two recursions of the three-way split formula and then we have extended this idea to an arbitrary number s of recursions.

In Table 3 we recall the complexity of the best known three-way approaches for the multiplication of binary polynomials of size $n = 3^k$. Specifically, we report the complexity results of Murat *et al.* [2] and of Fan *et al.* [4]. The proposed approach provides formulas and complexity results for any value of s. Here, we focus on three situations: the case $s = 2$ which assumes that $n = 3^{2k'}$ and has been studied in details in Section 3. We also provide the complexity for the case $s = 3$ under the assumption $n = 3^{3k'}$: the reported complexity in Table 3 is the non-recursive form of the following complexity derived from Lemma 4 for $s = 3$

$$\begin{cases} \mathcal{S}_\oplus(n) &= 216\mathcal{S}_\oplus(n/27) + \frac{368}{9} n - 242, \\ \mathcal{S}_\otimes(n) &= 216\mathcal{S}_\otimes(n/27), \\ \mathcal{D}(n) &= 10 D_\oplus + \mathcal{D}(n/27). \end{cases}$$

We also report in Table 3 the complexity of Corollary 1 when the optimization on the reconstruction is performed to the full recursion tree.

The results presented above show that the generalization of the Bernstein reconstruction to the case of three-way split formula reduce significantly the space complexity. The improvement can be seen in the leading terms of the number of bit additions $\mathcal{S}_\oplus(n)$: for $s = 1$ we have $4.8 n^{\log_3(6)} + O(n)$, the use of GBR$_2$ in the reconstruction provide $\mathcal{S}_\oplus(n) = 4.74\, n^{\log_3(6)} + O(n)$ and the use of GBR$_3$ gives $4.71\, n^{\log_3(6)} + O(n)$.

TABLE 3

Space and time complexities of three-way multiplier for $n = 3^k$

Method	# AND	# XOR	Delay$^{(*)}$
Murat *et al.* formula [2] (reviewed in Section 2)	$n^{\log_3(6)}$	$4.8n^{\log_3(6)} - 6n + 1.2$	$4\log_3(n)D_\oplus + D_\otimes$
Fan *et al.* overlap-free approach [4]	$n^{\log_3(6)}$	$5.33n^{\log_3(6)} - \frac{22n}{3} + 2$	$3\log_3(n)D_\oplus + D_\otimes$
Proposed $s = 2$ (cf. Section 3)	$n^{\log_3(6)}$	$4.74\,n^{\log_3(6)} - 5.89n + 1.14$	$3.5\log_3(n)D_\oplus + D_\otimes$
Proposed $s = 3$	$n^{\log_3(6)}$	$4.71\,n^{\log_3(6)} - 5.84n + 1.12$	$3.33\log_3(n)D_\oplus + D_\otimes$
Proposed $s = \log_3(n)$ (Corollary 1)	$n^{\log_3(6)}$	$4.68\,n^{\log_3(6)} - 2.2n - 0.4\log_3(n) + 0.82$	$3\log_3(n)D_\oplus + D_\otimes$

$(*)$ $D_\oplus$ represents the delay of an XOR gate and $D_\otimes$ represents the delay of an AND gate.

The case $s = \log_3(n)$ can be seen as the limit case, it has the smallest leading terms $4.68\,n^{\log_3(6)} + O(n)$ among all cases.

We also notice that the use of GBR$_s$ also improves the delay of the multiplier: for $s = 1$ we have $4\log_3(n)D_\oplus + D_\otimes$, for $s = 2$ we have $3.5\log_3(n)D_\oplus + D_\otimes$ and for $s = 3$ we have $3.33\log_3(n)D_\oplus + D_\otimes$. Again, the case $s = \log_3(n)$ can be seen as the limit case reaching the lowest delay $3\log_3(n)D_\oplus + D_\otimes$. This later case has a delay of the same order as the delay of the three-way multiplier of [4].

REFERENCES

[1] D. J. Bernstein. Batch Binary Edwards. In *Proceedings of Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 317–336. Springer, 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 and M. A. Hasan. A New Approach to Sub-quadratic Space Complexity Parallel Multipliers for Extended Binary Fields. *IEEE Trans. Computers*, 56(2):224–233, 2007.

[4] 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.

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

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

[7] T. Kohno, J. Viega, and D. Whiting. CWC: A High-Performance Conventional Authenticated Encryption Mode. In *FSE 2004*, volume 3017 of *LNCS*, pages 408–426. Springer, 2004.

[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. Negre. Efficient Binary Polynomial Multiplication Based on Optimized Karatsuba Reconstruction. Technical Report hal-00724778, Team DALI/LIRMM, on Hyper Articles en Ligne (HAL), 2012.

[11] NIST. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, November 2007.

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

[13] S. Winograd. *Arithmetic Complexity of Computations*. Society For Industrial & Applied Mathematics, U.S., 1980.