HAL
archives-ouvertes.fr

Efficient Subquadratic Space Complexity Binary Polynomial Multipliers Based On Block Recombination

Murat Cenk, Anwar Hasan, Christophe Negre

▶ To cite this version:

Murat Cenk, Anwar Hasan, Christophe Negre. Efficient Subquadratic Space Complexity Binary Polynomial Multipliers Based On Block Recombination. IEEE Transactions on Computers, Institute of Electrical and Electronics Engineers, 2014, 63 (9), pp.2273-2287. 10.1109/TC.2013.105 . hal-00712090v2

HAL Id: hal-00712090
https://hal.inria.fr/hal-00712090v2

Submitted on 7 Sep 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Efficient Subquadratic Space Complexity Binary Polynomial Multipliers Based On Block Recombination

M. Cenk, M.A. Hasan and C. Negre

Abstract

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

Index Terms

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

1 Introduction

Finite fields are part of important applications like coding theory and cryptography. For example, protocols over elliptic curves [10], [9] require several hundreds of multiplications and additions either in a prime or extended binary finite field of size $[2^{160}, 2^{600}]$. Efficient finite field arithmetic, specifically field multiplication and addition, is thus essential to obtain improved implementations of cryptographic protocols.

In this paper, we focus on bit parallel hardware implementation of multiplication over extended binary fields $\mathbb{F}_{2^n}$. Such a field can be defined as the set of univariate binary polynomials modulo an irreducible polynomial P of degree n. An addition of two elements in $\mathbb{F}_{2^n}$ consists of pairwise modulo-two addition of the coefficients, and this can be easily implemented with n parallel XOR gates. The multiplication in $\mathbb{F}_{2^n}$ consists of a regular binary univariate polynomial multiplication followed by a reduction modulo

M. Cenk and A. Hasan are with ECE Department and CACR, University of Waterloo, Waterloo, Ontario, Canada.

C. Negre is with UPVD,DALI, F-66860,Perpignan , France. and Univ. Montpellier II, LIRMM, UMR 5506, F-34095, Montpellier, France. and CNRS, LIRMM, UMR 5506, F-34095, Montpellier, France.

an irreducible polynomial P. When P is a sparse polynomial, i.e., a trinomial or a pentanomial, the reduction consists of few shifts and additions and is thus quite simple to implement. The multiplication of polynomials is generally more costly. For the considered range of n, the most efficient methods to multiply two binary polynomials are the subquadratic method based on Karatsuba [11] or three-way split formulas [12] which have a complexity of $O(n^{1+\varepsilon})$ bit operations with $0 < \varepsilon < 1$.

During the past few years improvements have been made on the Karatsuba and three-way split formulas. Specifically, Bernstein [2] and Zhou-Michalik [14] have independently improved the space complexity of Karatsuba formula by optimizing the reconstruction part of the formula. Bernstein in [2] has also extended this approach to two recursions of the Karatsuba formula, leading to a four-way split formula with a better space requirement than the original Karatsuba formula. In [4], Cenk *et al.* have applied the technique of Bernstein to improve the space complexity of the three-way split formula with six recursive multiplications. Finally, Fan *et al.* [6] have optimized the delay of two and three-way split formulas by using a different kind of splitting of the polynomials.

In this paper we propose some further optimizations of three and four-way formulas. Concerning the three-way split, we remove some redundant computation in the formula of [4] to reduce the space complexity. We then combine this idea with the approach of [6] to obtain a three-way split formula which has the same delay as the one in [6] but with a smaller space complexity. For the four-way formula, we combine the four-way approach of [2] and the modified splitting of [6], this yields a formula which has the same delay as [6] but with a lower space complexity.

We also investigate an extension of the block recombination approach proposed in [8] which reduces the space complexity of a subquadratic space complexity architecture for Toeplitz matrix-vector product. Since subquadratic approaches for Toeplitz matrix-vector product and subquadratic algorithms for polynomial multiplication are very similar, we can extend the approach of Toeplitz matrix-vector product to the considered two-,three- and four-way split multiplier. This extension needs to reevaluate the different cost of the different blocks of the multiplier. Then we will be able to recombine a parallel architecture performing two polynomial multiplications in parallel followed by an addition and then apply this recombination to different two-, three- and four-way split multipliers. This results in multipliers with smaller space complexities while having the same delay.

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

2 REVIEW OF SUBQUADRATIC METHODS FOR POLYNOMIAL MULTIPLICATIONS

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

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

This method, known as the schoolbook method, requires n^2 bit multiplications and $(n-1)^2$ bit additions. When $n \in [160, 600]$ and the bit operations are performed in parallel fashion, this approach results in a quite large circuit. For this size of n, subquadratic methods are generally more suitable for practical applications. These methods split each polynomial in two or three and then express the product C in terms of a number of products of smaller degrees. When this process is applied recursively, it results in $O(n^\delta),\ \delta \in \{\log_2(3), \log_3(6)\}$, bit operations for the whole multiplication. In this section we review the best known two and three-way split approaches.

Remark 1: In the sequel, we will not discuss the reduction operation modulo the irreducible polynomial which defines the finite field $\mathbb{F}_{2^n}$. We just mention that when P has a sparse form, i.e., for example P is a trinomial $P = X^n + X^k + 1$ with $k < n/2$, then the reduction can be performed by only a few shifts and additions. Indeed let $C = \sum_{i=0}^{2n-2} c_i X^i$ be a product of two degree $n-1$ polynomials then:

$$C = \left(\sum_{i=0}^{n-1} c_i X^i \right) + \left(\sum_{i=n}^{2n-2} c_i X^{i-n} \right) \left(X^k + 1 \right) \mod P$$

since $X^n \equiv X^k + 1 \mod P$. This means that after applying this process twice we would get a polynomial of degree $< n$.

2.1 Review of two-way formulas

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

2.1.1 Karatsuba formula

We consider two n-term $A(X)$ and $B(X)$ in $\mathbb{F}_2[X]$ where n is supposed to be a power of 2. The original Karatsuba approach consists of splitting A and B in two parts

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

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

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

$$P_0 = A_0 B_0, \qquad P_1 = A_1 B_1, \qquad P_2 = (A_0 + A_1)(B_0 + B_1).$$

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

$$C = P_0 + (P_0 + P_1 + P_2)X^{n/2} + P_1X^n.$$

Following [2], [14] we perform this reconstruction in three steps:

$$R_0 = P_0 + X^{n/2}P_1, \quad R_1 = R_0(1 + X^{n/2}), \quad C = R_1 + P_2X^{n/2},$$

and this requires $5n/2 - 3$ bit operations.

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

$$\begin{cases} \mathcal{S}_\oplus(n) &= 7n/2 - 3 + 3\mathcal{S}_\oplus(n), \\ \mathcal{S}_\otimes(n) &= 3\mathcal{S}_\otimes(n), \\ \mathcal{D}(n) &= 3D_\oplus + \mathcal{D}(n/2). \end{cases} \qquad \begin{cases} \mathcal{S}_\oplus(n) &= \mathcal{S}_\oplus(n) = \frac{11}{2}n^{\log_2(3)} - 7n + \frac{3}{2}, \\ \mathcal{S}_\otimes(n) &= n^{\log_2(3)}, \\ \mathcal{D}(n) &= 3\log_2(n)D_\oplus + D_\otimes. \end{cases} \tag{1}$$

Remark 2: The Karatsuba approach applies over more general fields or rings: in those more general cases the formula is the same as the one previously reported except for the reconstruction part $C = P_0 + (P_2 - P_0 - P_1)X^{n/2} + P_1X^n$ which involves two subtractions. Since in $\mathbb{F}_2$ we have $-1 = 1$ these minus signs disappear in the case of binary polynomials. This remark also applies to the others formulas reported in the remainder of this paper.

2.1.2 Overlap free method for two-way split multiplication

Fan *et al.* in [6] use a different way to split the polynomials A and B in the Karatsuba formula leading to a reduction in the delay of the resulting parallel multiplier. Indeed, following [6], we separate the even and odd coefficients of A and B as follows

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

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

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

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

- *Reconstruction.* We then reconstruct C as

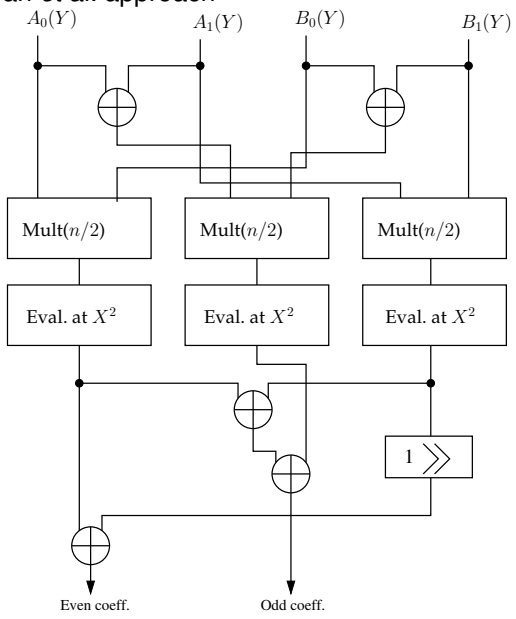
$$C = P_0(X^2) + \Big(P_0(X^2) + P_1(X^2) + P_2(X^2)\Big)X + P_1(X^2)X^2. \tag{3}$$

The space requirement ($\mathcal{S}_\oplus(n)$ and $\mathcal{S}_\otimes(n)$) is slightly larger than the former Karatsuba formula (1),

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

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

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



2.2 Review of three-way multiplication

In this subsection, we review the best known three-way split formulas with six recursive multiplications. Specifically, we review the formula proposed in [4] which has the smallest space complexity and the formula of Fan *et al.* [6] which has the smallest delay among all formulas. As before $A(X)$ and $B(X)$ are two binary polynomials of degree $n-1$ in $\mathbb{F}_2[X]$ where n is a power of 3.

- The formula based on the regular splitting [4] decompose these two polynomials $A(X)$ and $B(X)$ in three parts and replace $X^{n/3}$ by a new indeterminate Y. This results in two polynomials $A(Y) = A_0 + A_1Y + A_2Y^2$ and $B(Y) = B_0 + B_1Y + B_2Y^2$ of degree two in Y. We can then use formula for multiplication of degree two polynomials to derive the formula shown in Subtable 1(a).
- Fan *et al.* in [6] have proposed to modify the three-way split formula by using a different splitting.

Specifically, the authors in [6] split the polynomials A and B in three parts as

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

Then, we can view $A_i(X^3)$ and $B_i(X^3), i = 0, 1, 2,$ as degree $n/3-1$ polynomials in Y evaluated at X^3 and we consider $A = A_0(Y) + A_1(Y)X + A_2(Y)X^2$ and $B = B_0(Y) + B_1(Y)X + B_2(Y)X^2$ as polynomials of degree two in X. Fan *et al.* in [6] have then applied formula for degree two polynomial and have derived the formula described in Subtable 1(b).

TABLE 1

Three-way split formula with six multiplications

(a) Regular splitting [4]

Rec. Prod.	$P_0 = A_0B_0$ $P_1 = A_1B_1$ $P_2 = A_2B_2$ $P_3 = (A_0 + A_1)(B_0 + B_1)$ $P_4 = (A_0 + A_2)(B_0 + B_2)$ $P_5 = (A_1 + A_2)(B_1 + B_2)$
Reconst.	$R_0 = (P_0 + X^{n/3}P_1 + X^{2n/3}P_2)$ $R_1 = R_0(1 + X^{n/3} + X^{2n/3})$ $C = R_1 + P_3X^{n/3} + P_4X^{2n/3}$ $+P_5X^{3n/3}$
Compl.	$\mathcal{S}_\oplus(n) = \frac{79}{15}n^{\log_3(6)} - \frac{20n}{3} + \frac{7}{5}$ $\mathcal{S}_\otimes(n) = n^{\log_3(6)}$ $\mathcal{D}(n) = 4\log_3(n)D_\oplus + D_\otimes$

(b) Overlap free splitting [6]

Rec. Prod.	$P_0(Y) = A_0(Y)B_0(Y)$ $P_1(Y) = A_1(Y)B_1(Y)$ $P_2(Y) = A_2(Y)B_2(Y)$ $P_3(Y) = (A_0(Y) + A_1(Y))(B_0(Y) + B_1(Y))$ $P_4(Y) = (A_0(Y) + A_2(Y))(B_0(Y) + B_2(Y))$ $P_5(Y) = (A_1(Y) + A_2(Y))(B_1(Y) + B_2(Y))$
Reconst.	$R_0 = P_0(X^3) + X^3\left(P_5(X^3) + P_1(X^3) + P_2(X^3)\right)$ $R_1 = P_3(X^3) + P_0(X^3) + P_1(X^3) + X^3P_2(X^3)$ $R_2 = P_4(X^3) + P_0(X^3) + P_1(X^3) + P_2(X^3)$ $C = R_0 + XR_1 + X^2R_2$
Compl.	$\mathcal{S}_\oplus(n) = \frac{16}{3}n^{\log_3(6)} - \frac{22n}{3} + 2$ $\mathcal{S}_\otimes(n) = n^{\log_3(6)}$ $\mathcal{D}(n) = 3\log_3(n)D_\oplus + D_\otimes$

We have also reported in Table 1 the resulting overall cost, in terms of bit addition $S_\oplus(n)$ and bit multiplication $S_\otimes(n)$ of the two three-way split formulas, along with delay of the multipliers. These complexity results show that in term of space complexity the formula of [4] is the best choice. The formula of [6] have better delay due to the interleaving form of the reconstruction part of the formulas.

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

In [2], Bernstein has shown that it is possible to obtain further improvement in polynomial multiplication when $n = 2^k$ by considering a four-way split formula. We review here this approach. Let A and B be two polynomials of degree $n-1$, which are split in four parts $A = A_0 + A_1X^{n/4} + A_2X^{2n/4} + A_3X^{3n/4}$ and $B = B_0 + B_1X^{n/4} + B_2X^{2n/4} + B_3X^{3n/4}$, where A_i and B_i have degree $n/4-1$ for $i = 0, 1, 2, 3$. The idea of Bernstein is to apply two recursions of the two-way split formula and then optimize the reconstruction

part of the resulting formula. This results in nine recursive products of polynomials of size $n/4$:

$$\begin{array}{lll} P_0 = A_0B_0, & P_1 = A_1B_1 & P_2 = (A_0 + A_1)(B_0 + B_1), \\ P_3 = A_2B_2, & P_4 = A_3B_3, & P_5 = (A_2 + A_3)(B_2 + B_3), \\ P_6 = (A_0 + A_2)(B_0 + B_2) & P_7 = (A_1 + A_3)(B_1 + B_3), & P_8 = (A_0 + A_1 + A_2 + A_3)(B_0 + B_1 + B_2 + B_3). \end{array}$$

Following [2], we apply two recursions of the Karatsuba reconstruction which results in the following expression of C in terms of $P_i, i = 0, \ldots, 8$.

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

Bernstein has proposed to arrange the previous expression as follows

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

As noticed by Bernstein, the above formula requires $n/4 - 1$ fewer bit additions than two recursions of the Karatsuba formula (Subsection 2.1.1). The recursive expression of the delay of Bernstein's four-way multiplier is $\mathcal{D}(n) = 5D_\oplus + \mathcal{D}(n/4)$, and this is better by one $D_\oplus$ than the delay of two recursions of Karatsuba formula. The resulting non-recursive expression of the four-way split formula, when n is an even power of 2, is as follows

$$\left\{ \begin{array}{rcl} S_\oplus(n) &=& \frac{217}{40}n^{\log_2(3)} - \frac{34n}{5} + \frac{11}{8}, \\ S_\otimes(n) &=& n^{\log_2(3)}, \\ \mathcal{D}(n) &=& \frac{5}{2}\log_2(n)D_\oplus + D_\otimes. \end{array} \right. \quad (5)$$

3 PROPOSED OPTIMIZATION FOR THREE AND FOUR-WAY FORMULAS

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

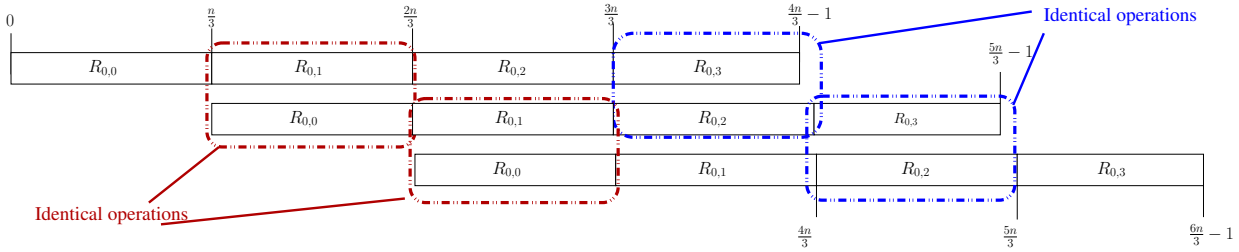
3.1 Improvement on the space complexity of three-way multiplication

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

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

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

Fig. 2. Redundant computations in three-way reconstruction



We can modify the reconstruction in order to save each of these re-occurring two term sums. For this, we split the three products P_0, P_1 and P_2 in two parts $P_0 = P_{0,L} + X^{n/3}P_{0,H}, P_1 = P_{1,L} + X^{n/3}P_{1,H}$ and $P_2 = P_{2,L} + X^{n/3}P_{2,H}$ where $P_{i,L}$ and $P_{i,H}$ are degree $n/3-2$ polynomials for $i = 0, 1, 2$. We then express the terms $R_{0,i}$ of $R_0 = R_{0,0} + X^{n/3}R_{0,1} + X^{2n/3}R_{0,2}$ and R_1 in terms of $P_{i,L}$ and $P_{i,H}$ for $i = 0, 1, 2$ and the terms $R_{1,i} = R_{1,0} + X^{n/3}R_{1,1} + X^{2n/3}R_{1,2} + X^{3n/3}R_{1,3} + X^{4n/3}R_{1,4}$ of R_1 in terms of $R_{0,i}$ and $P_{i,L}$ and $P_{i,H}$. The last computation for C remains unchanged. The resulting modified reconstruction formula is given in Table 2.

TABLE 2

Space efficient three-way split reconstruction

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

If we consider the overall optimized formula which comprises the recursive products of Subtable 1(a) plus the proposed optimized reconstruction of Table 2, we obtain the recursive and non-recursive forms

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

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

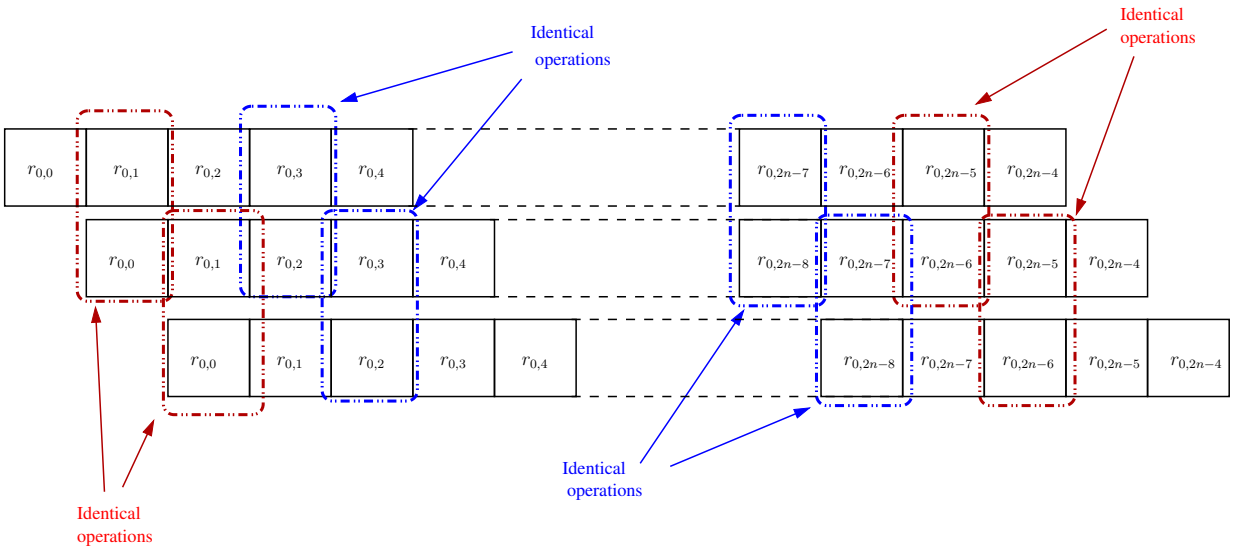
3.2 Improvement of the overlap free three-way formula

In this subsection, we present an optimization of the overlap free three-way split formula (Subtable 1(b)). We do by leaving the recursive multiplications of the formula unchanged. Let $P_0(Y),\ldots,P_5(Y)$ be the six products. We solely modify the sequence of operations in the reconstruction: we propose to perform the reconstruction of C as follows

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

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

Fig. 3. Redundant computations in overlap free three-way reconstruction



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

The resulting optimized bit operations for R_1 are given below:

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

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

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

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

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

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

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

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

3.3 Proposed four-way split formula

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

four-way formula of Bernstein and the overlap free method of [6]. Indeed, we use the following special splitting for the two polynomials A and B of degree $n-1$:

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

where $A_i(Y) = \sum_{j=0}^{n/4-1} a_{i+4j}Y^j$ and $B_i(Y) = \sum_{j=0}^{n/4-1} b_{i+4j}Y^j$ are each a polynomial of degree $n/4-1$ in Y. Then we can consider $A = A_0(Y) + XA_1(Y) + X^2A_2(Y) + X^3A_3(Y)$ and $B = B_0(Y) + XB_1(Y) + X^2B_2(Y) + X^3B_3(Y)$ as polynomials of degree 3 in X. We can then apply the four-way split formula of Bernstein (Subsection 2.3) to compute the product. To obtain the expression of $C(X) = A(X) \times B(X)$ we need to replace Y with X^4, this is done just before proceeding to the reconstruction:

$$\begin{aligned} C &= \Big(\big(P_0(X^4) + XP_1(X^4) + X^2P_3(X^4) + X^3P_4(X^4)\big)\big(1+X\big) + XP_2(X^4) + X^3P_5(X^4)\Big)(1+X^2) \\ &\quad + \Big(\big(P_6(X^4) + XP_7(X^4)\big)(1+X) + P_8(X^4)X\Big)X^2. \end{aligned}$$

We provide a detailed four-way split formula in Table 3.

TABLE 3

Proposed four-way split formulas with nine recursive multiplications

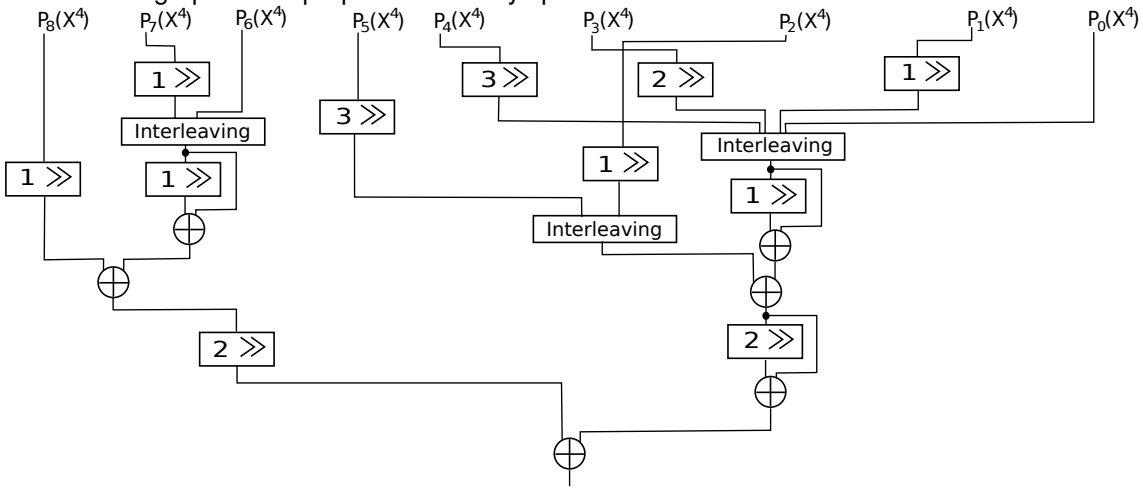
Operations	Computations	Cost		
		$\#\oplus$	$\#\otimes$	Delay
Rec. Prod.	$P_0(Y) = A_0B_0$	$S_\oplus(n/4)$	$S_\otimes(n/4)$	$\mathcal{D}(n/4)$
	$P_1(Y) = A_1B_1$	$S_\oplus(n/4)$	$S_\otimes(n/4)$	$\mathcal{D}(n/4)$
	$P_2(Y) = (A_0+A_1)(B_0+B_1)$	$S_\oplus(n/4)+n/2$	$S_\otimes(n/4)$	$D_\oplus + \mathcal{D}(n/4)$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

Space complexity. The costs of the recursive products are unchanged compared to Subsection 2.3: only the costs of the steps of the reconstruction, which are different, are to be determined. In Table 3 we detail

these costs and derive the recursive form of the overall complexity. The corresponding non-recursive form of the complexity is given below:

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

Fig. 4. Data flow graph of the proposed four-way split reconstruction



Time complexity. The data-flow of the reconstruction part of the proposed four-way formula is depicted in Fig. 4. In this data-flow graph the *Interleaving* blocks and shifting blocks do not involve any logic gate. A close look at this data-flow graph shows that any path starting from the input A or B of the multiplier and goes through any of the P_i and then down to C encounter, at most, four XOR gates. We deduce that the critical path delay is $\mathcal{D}(n) = 4D_{\oplus} + \mathcal{D}(n/4)$ which results in the non-recursive form $\mathcal{D}(n) = 4\log_4(n)D_{\oplus} + D_{\otimes} = 2\log_2(n)D_{\oplus} + D_{\otimes}$ for n an even power of 2.

4 Block recombination for polynomial multiplication

In this section we present a block recombination approach which reduces the space complexity of the multipliers presented in Sections 2 and 3. This approach is an extension of the approach presented in [8] for Toeplitz matrix-vector multiplication (TMVP). Indeed, the subquadratic formula for TMVP and polynomial multiplication are very similar; they differ slightly since there are no computations corresponding to the computations on the Toeplitz matrix and the reconstruction computations are different. We thus follow the same process as in [8]:

- We decompose the parallel multipliers of Sections 2 and 3 into a number of blocks.
- We then adapt the optimization of [8] concerning two-multiplications-and-add architecture to the case of polynomial multiplication.

- We then apply this optimization to recombine the block of the considered multipliers. This recombination is slightly different from the case of TMVP since the blocks involved are different: there are fewer blocks for the component formation and more blocks in the reconstruction than for TMVP computation.

4.1 Decomposition of polynomial multiplication algorithms

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

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

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

  We can similarly define $CPF_{2,n}$ and $CPF_{3,n}$ functions for overlap free formulas: this requires us to replace the regular two-way and three-way splitting by the corresponding overlap-free splitting.
  In the sequel, we will often refer to the *component representation* of a degree $n-1$ polynomial A as the bit array $\hat{A} = CPF_{2,n}(A)$ (resp. $\hat{A} = CPF_{3,n}(A)$) which has a bit length of $n^{\log_2(3)}$ (resp. $n^{\log_3(6)}$).
- *Component multiplication (CM).* The component multiplication consists of the bit-wise multiplication of the component polynomial formations $\hat{A}$ and $\hat{B}$ of two polynomials A and B

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

  This operation can be implemented with $n^{\log_2(3)}$ parallel AND gates in the case of two-way split approaches and $n^{\log_3(6)}$ parallel AND gates in the case of three-way split approaches.
- *Reconstruction (R).* The reconstruction $R(\hat{C})$ of a vector $\hat{C}$ of length $n^{\log_2(3)}$ (resp. $n^{\log_3(6)}$) consists of recursively applying the reconstruction part of the considered two or four-way (resp. three-way) split formula of Subsections 2.1 , 2.3 (resp. Subsections 2.2 or 3.1) to obtain a polynomial C of degree $2n-2$. For example, the function R of the Karatsuba formula of Subsection 2.1.1 and the three-way

split formula of Subtable 1(a) are as follows

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

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

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

In the sequel, we will recombine the blocks of the multiplier in order to reduce the space complexity. But first, we establish the complexity of each block of the multiplier for various formulas presented in Sections 2 and 3.

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

Two and four-way formulas	Three-way formulas
$\mathcal{S}^{CPF}_{2,\oplus}(n) = n^{\log_2(3)} - n$	$\mathcal{S}^{CPF}_{3,\oplus}(n) = n^{\log_3(6)} - n$
$\mathcal{S}^{CM}_{2,\otimes}(n) = n^{\log_2(3)}$	$\mathcal{S}^{CM}_{3,\otimes}(n) = n^{\log_3(6)}$
$\mathcal{S}^{R}_{2,\oplus}(n) = \mathcal{S}_{\oplus}(n) - 2(n^{\log_2(3)} - n)$	$\mathcal{S}^{R}_{3,\oplus}(n) = \mathcal{S}_{\oplus}(n) - 2(n^{\log_3(6)} - n)$

Proof: For the complexity of the CPF function, from (9), we have

$$\begin{aligned} \mathcal{S}^{CPF}_{2,\oplus}(n) &= n/2 + 3\mathcal{S}^{CPF}_{2,\oplus}(n/2) \quad \Rightarrow \quad \mathcal{S}^{CPF}_{2,\oplus}(n) = n^{\log_2(3)} - n, \\ \mathcal{S}^{CPF}_{3,\oplus}(n) &= 3n/3 + 6\mathcal{S}^{CPF}_{3,\oplus}(n/3) \quad \Rightarrow \quad \mathcal{S}^{CPF}_{3,\oplus}(n) = n^{\log_3(6)} - n, \end{aligned}$$

as stated in Lemma 1. For the CM function, it is a direct consequence of the fact that a component element $\hat{A}$ has a bit length of $n^{\log_2(n)}$ (resp. $n^{\log_3(6)}$). For the complexity of the R function: it is equal to the complexity of the multiplier minus the complexity of two CPFs and one CM. □

Explicit complexity based on the different formula stated in Sections 2 and 3 are reported in Table 4. We remark that the repartition of the gate counts into the blocks CPF, CM and R is slightly different than the repartition into the blocks CMF, CVF, CM and R of the TMVP multiplier [8]. Specifically, here, there is one big block R while the three others are smaller, but in the case of TVMP there were two big blocks CMF and R, and two small blocks CVF and CM.

4.2 Block recombination of two-multiplications-and-add

In this subsection, we state the basic block recombination operation, which we will use later to recombine the two and three-way multipliers. Specifically, we consider the problem of performing two instances of

TABLE 4

Space complexity of the reconstruction blocks

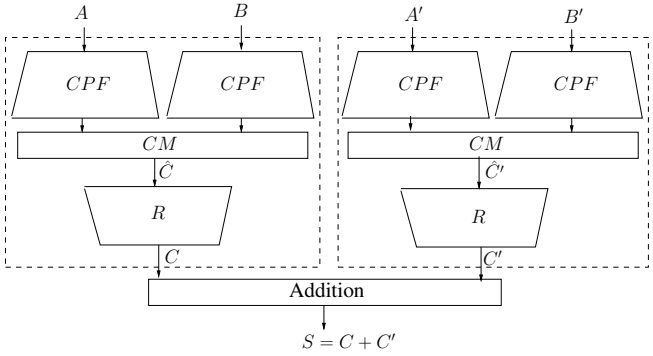
Method	Split	Complexity		
		$\mathcal{S}_{\oplus}^{CPF}(n)$	$\mathcal{S}_{\otimes}^{CM}(n)$	$\mathcal{S}_{\oplus}^{R}(n)$
Subsection 2.1.1	2	$n^{\log_2(3)} - n$	$n^{\log_2(3)}$	$\frac{7}{2}n^{\log_2(3)} - 5n + \frac{3}{2}$
Subsection 2.1.2	2	$n^{\log_2(3)} - n$	$n^{\log_2(3)}$	$4n^{\log_2(3)} - 6n + 2$
Subtable 1(a)	3	$n^{\log_3(6)} - n$	$n^{\log_3(6)}$	$\frac{49}{15}n^{\log_3(6)} - \frac{14n}{3} + \frac{7}{5}$
Subtable 1(b)	3	$n^{\log_3(6)} - n$	$n^{\log_3(6)}$	$\frac{10}{3}n^{\log_3(6)} - \frac{16n}{3} + 2$
Subtable 3.1	3	$n^{\log_3(6)} - n$	$n^{\log_3(6)}$	$\frac{14}{5}n^{\log_3(6)} - 4n + \frac{6}{5}$
Subsection 3.2	3	$n^{\log_3(6)} - n$	$n^{\log_3(6)}$	$\frac{16}{5}n^{\log_3(6)} - 5n + \frac{9}{5}$
Subsection 2.3	4	$n^{\log_2(3)} - n$	$n^{\log_2(3)}$	$\frac{137}{40}n^{\log_2(3)} - \frac{24}{5}n + \frac{11}{8}$
Subsection 3.3	4	$n^{\log_2(3)} - n$	$n^{\log_2(3)}$	$\frac{31}{8}n^{\log_2(3)} - 6n + \frac{17}{8}$

polynomial multiplications in parallel followed by an addition. In other words, if A, A', B and B' are polynomials of degree $n-1$ each, we want to compute

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

A straightforward approach to design a parallel architecture which computes S is shown in Fig. 5. In

Fig. 5. Two-multiplications-and-an-add architecture



order to reduce the space complexity of this architecture, we recombine the blocks of Fig. 5 based on the following lemma.

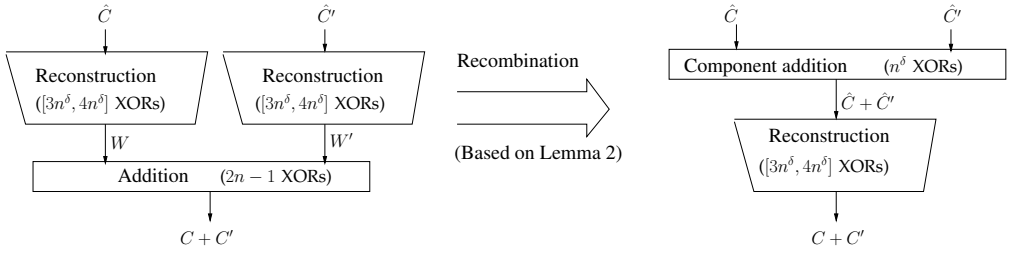
Lemma 2: Let $\hat{C}$ and $\hat{C}'$ be two arrays of $n^{\log_2(3)}$ bits (resp. $n^{\log_3(6)}$ bits). Let $R_{2,n}$ (resp. $R_{3,n}$) be the reconstruction function, then we have

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

The proof is very similar to the proof of Property 1 in [8] in the case of TMVP: the only difference is that we have a reconstruction formula which is slightly different. We thus just skip this proof here.

The previous lemma shows that the sequence of operations, *two-reconstructions-then-add*, can be reversed, i.e., we can first perform an addition of the original input components and then perform the reconstruction on the sum. In Fig. 6, we apply this property to recombine the lower two layers of blocks of the two-multiplication-and-an-add architecture (Fig. 5). We call an addition of two components $\hat{C}$ and $\hat{C}'$ a component addition (CA). This CA can be computed with $n^\delta, \delta \in \{\log_2(3), \log_3(6)\}$ parallel XOR gates. We thus have placed a block of this kind in the right side of Fig. 6.

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



In Fig. 6 we also provide the sizes of different blocks: an addition of input components (CA) requires n^δ XOR gates, where $\delta \in \{\log_2(3), \log_3(6)\}$. Similarly, we note that the reconstruction block has a space complexity in the interval $[3n^\delta, 4n^\delta]$ (see Table 4). The recombination replaces a block consisting of at least $3n^\delta$ XOR gates by a block that has only n^δ XOR gates. This means that we save at least $2n^\delta$ XOR gates.

4.3 Two-way split block recombination

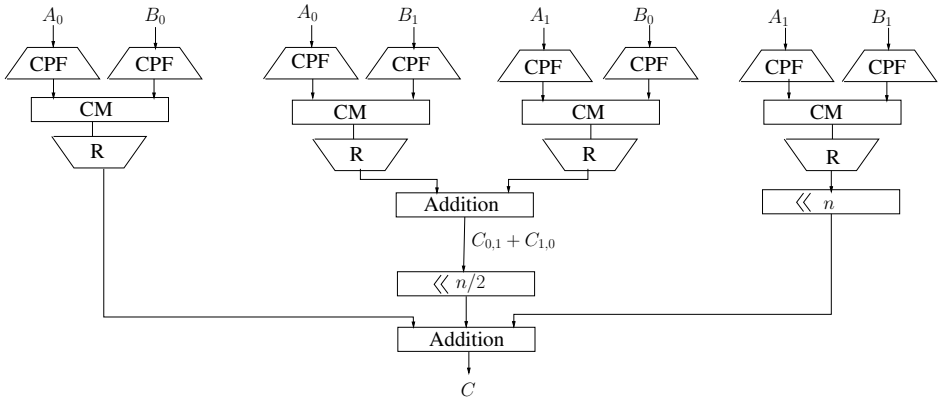
In this subsection we present a block recombination approach for the two-way split multiplier in order to reduce the space complexity. Let A and B be two degree $n-1$ polynomials where $n > 1$ is a power of 2. We split A and B in two parts $A = A_0 + A_1 X^{n/2}$ and $B = B_0 + B_1 X^{n/2}$, where A_i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

$$AB = A_0 B_0 + (A_0 B_1 + A_1 B_0) X^{n/2} + A_1 B_1 X^n. \tag{13}$$

We can design an architecture based on the previous expression of $C = AB$ which independently performs each product $C_{i,j} = A_i B_j$ through a two-way split multiplier. After that we reconstruct C by performing the additions and shifts corresponding to the expression $C = C_{0,0} + (C_{0,1} + C_{1,0}) X^{n/2} + C_{1,1} X^n$. The resulting architecture is depicted in Fig. 7.

We then note that the computation of $C_{0,1} + C_{1,0}$ in Fig. 7 consists of two parallel multiplications followed by an addition. We can apply the block recombination presented in Subsection 4.2 which reverses the order of the reconstruction and the addition. We also notice that we can remove some redundant

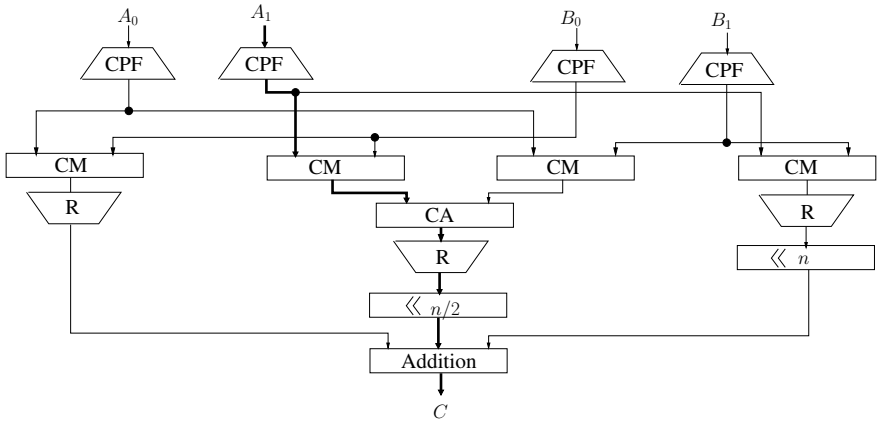
Fig. 7. First step in the recombination of two-way split multiplier



blocks $CPF_{2,n/2}(A_i)$, $i = 0, 1$ and $CPF_{2,n/2}(B_j)$, $j = 0, 1$ by keeping only one $CPF_{2,n/2}$ block for each A_i and B_j for $i = 0, 1$ and $j = 0, 1$.

This results in the architecture shown in Fig. 8.

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



Complexity of the recombined architecture (Fig 8). We first evaluate the space complexity of the recombined architecture. We note that the final addition block requires $n - 2$ XOR gates since the degree of the polynomials $C_{0,0}, C_{0,1}$ and $C_{1,0}$ is at most $2n/2 - 2$. We then express the space complexity in terms of the complexity of each block:

$$\begin{cases} \mathcal{S}_{\oplus}(n) &= 4\mathcal{S}_{\oplus}^{CPF}(n/2) + 3\mathcal{S}_{\oplus}^{R}(n/2) + \mathcal{S}_{\oplus}^{CA}(n/2) + (n-2), \\ \mathcal{S}_{\otimes}(n) &= 4\mathcal{S}_{\otimes}(n/2). \end{cases} \tag{14}$$

In (14), we can replace the terms $\mathcal{S}_{\oplus}^{CPF}(n/2), \mathcal{S}_{\oplus}^{R}(n/2), \mathcal{S}_{\oplus}^{CA}(n/2)$ and $\mathcal{S}_{\otimes}(n/2)$ by their explicit expressions in terms of n given in Lemma 1 and Table 4. This leads to the complexity results reported in the

middle two columns of Table 5.

TABLE 5

Complexity of two-way recombined multipliers

Recombined formula	# AND	# XOR	Delay
Two-way of [2], [14]	$\frac{4}{3}n^{\log_2(3)}$	$\frac{31}{6}n^{\log_2(3)} - \frac{17n}{2} + \frac{5}{2}$	$(3\log_2(n)-1)D_\oplus + D_\otimes$
Two-way of [6]	$\frac{4}{3}n^{\log_2(3)}$	$\frac{17}{3}n^{\log_2(3)} - 10n + 4$	$2\log_2(n)D_\oplus + D_\otimes$
Four-way of Subsection 2.3	$\frac{4}{3}n^{\log_2(3)}$	$\frac{611}{120}n^{\log_2(3)} - \frac{41n}{5} + \frac{17}{8}$	$(\frac{5}{2}\log_2(n) - \frac{1}{2})D_\oplus + D_\otimes$
Four-way of Subsection 3.3	$\frac{4}{3}n^{\log_2(3)}$	$\frac{133}{24}n^{\log_2(3)} - 10n + \frac{35}{8}$	$2\log_2(n)D_\oplus + D_\otimes$

We now evaluate the delay of the two-way split multiplier obtained through block recombination. The critical path is shown in bold in Fig. 8. Its delay consists of $2D_\oplus$ for the additions (the CA block and the final Addition block), plus the delay of the blocks CPF, CA and R

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

We now remark that $\mathcal{D}^{CPF}(n/2) + \mathcal{D}^{CM}(n/2) + \mathcal{D}^{R}(n/2)$ is equal to the delay $\mathcal{D}(n/2)$ of a regular two-way split multiplier of size $n/2$. This results in the delays reported in the right most column of Table 5 corresponding to the formulas presented in Sections 2 and 3.

4.4 Four-way and three-way recombination

In the previous subsection, we have presented a two-way block recombination. We sketch here two extensions of this approach:

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

$$AB = \underbrace{A_0B_0}_{C_0} + \underbrace{(A_0B_1 + A_1B_0)}_{C_1}X^{n/3} + \underbrace{(A_0B_2 + A_1B_1 + A_2B_0)}_{C_2}X^{2n/3} + \underbrace{(A_1B_2 + A_2B_1)}_{C_3}X^{3n/3} + \underbrace{A_2B_2}_{C_4}X^{4n/3}. \tag{15}$$

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

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

Based on the above three-way (resp. four-way) expressions of AB, we can design a three-way (resp. four-way) block recombined multiplier as follows:

1) We apply the CPF function on each polynomials A_i and B_i.
2) We apply the CM function $CPF(A_i) \otimes CPF(B_j)$ for all i, j in parallel.
3) The addition corresponding to each C_i are performed in component representation.
4) Each C_i is then reconstruct through a R block.
5) The final result C is finally obtain after performing the overlapping addition of (15) (resp. (16)).

We express the space complexity in terms of the complexities of the different blocks CPF, CM, CA and R plus the cost of the additions in the final step. We also derive the delay of the multiplier by finding the corresponding critical path of the considered multiplier. These complexities are given in Table 6.

TABLE 6

Complexities of the three-way and four-way recombined multipliers

.

Three-way block recomb.	Four-way block recomb.
$\mathcal{S}_{\oplus}(n)$=$6\mathcal{S}^{CPF}_{3,\oplus}(n/3) + 5\mathcal{S}^{R}_{3,\oplus}(n/3) + 4\mathcal{S}^{CA}_{3,\oplus}(n/3)$ $+4n/3 - 4$, $\mathcal{S}_{\otimes}(n)$=$9\mathcal{S}^{CM}_{3,\otimes}(n/3)$, $\mathcal{D}(n)$=$\underbrace{\mathcal{D}^{CPF}_{3}(n/3) + \mathcal{D}^{CM}_{3}(n/3) + \mathcal{D}^{R}_{3}(n/3)}_{(*)} + 3D_{\oplus}$.	$\mathcal{S}_{\oplus}(n)$=$8\mathcal{S}^{CPF}_{2,\oplus}(n/4) + 7\mathcal{S}^{R}_{2,\oplus}(n/4) + 9\mathcal{S}^{CA}_{2,\oplus}(n/2)$ $+3n/2 - 6$, $\mathcal{S}_{\otimes}(n)$=$16\mathcal{S}^{CM}_{2,\otimes}(n/4)$, $\mathcal{D}(n)$=$3D_{\oplus} + \underbrace{\mathcal{D}^{CPF}_{2}(n/4) + \mathcal{D}^{CM}_{2}(n/4) + \mathcal{D}^{R}_{2}(n/4)}_{(*)}$.

In the above table $(*)$ is equal to the delay of a three-way (resp. four-way) multiplier with input size $n/3$ (resp. $n/4$) bits. If we now replace the block complexities given in Lemma 1 and Table 4 in the above expressions, we obtain the complexities summarized in Table 7.

TABLE 7

Complexity of three-way recombined multiplier

	Method	# AND	# XOR	Delay
Three-way	Space optimized formula [4] (Subtable 1(a))	$\frac{9}{6}n^{\log_3(6)}$	$\frac{79}{18}n^{\log_2(3)} - \frac{76n}{9} + 3$	$(4\log_3(n) - 1)D_{\oplus} + D_{\otimes}$
	Delay optimized formula [6] (Subtable 1(b))	$\frac{9}{6}n^{\log_3(6)}$	$\frac{40}{9}n^{\log_2(3)} - \frac{86n}{9} + 6$	$3\log_3(n)D_{\oplus} + D_{\otimes}$
	Proposed space optimized formula (Subsection 3.1)	$\frac{9}{6}n^{\log_3(6)}$	$4n^{\log_3(6)} - \frac{22n}{3} + 2$	$(4\log_3(n) - 1)D_{\oplus} + D_{\otimes}$
.	Proposed delay optimized formula (Subsection 3.2)	$\frac{9}{6}n^{\log_3(6)}$	$\frac{13}{3}n^{\log_3(6)} - 9n + 5$	$3\log_3(n)D_{\oplus} + D_{\otimes}$
Four-way	Two-way of [2], [14] (Subsection 2.1.1)	$\frac{16}{9}n^{\log_2(3)}$	$\frac{83}{18}n^{\log_2(3)} - \frac{37n}{4} + \frac{9}{2}$	$(3\log_2(n) - 3)D_{\oplus} + D_{\otimes}$
	Two-way of [6] (Subsection 2.1.2)	$\frac{16}{9}n^{\log_2(3)}$	$5n^{\log_2(3)} - 11n + 8$	$(2\log_2(n) - 1)D_{\oplus} + D_{\otimes}$
	Four-way of [2] (Subsection 2.3)	$\frac{16}{9}n^{\log_2(3)}$	$\frac{1639}{360}n^{\log_2(3)} - \frac{89n}{10} - \frac{29}{8}$	$(\frac{5}{2}\log_2(n) - 2)D_{\oplus} + D_{\otimes}$
	Proposed four-way in Subsection 3.3	$\frac{16}{9}n^{\log_2(3)}$	$\frac{353}{72}n^{\log_2(3)} - 11n + \frac{71}{8}$	$(2\log_2(n) - 1)D_{\oplus} + D_{\otimes}$

Remark 3: The four-way block recombination is a particular case of 2^{ℓ}-way block recombination. We have evaluated the resulting complexities for various values of ℓ, and have noticed that the least total

gate count (AND and XOR combined) is obtained when $\ell = 2$. So the above four-way approach is the best approach when considering space complexity.

5 Complexity comparison and conclusion

In this section, we provide comparisons of the proposed formulas with best known approaches for binary field multiplication: formulas for polynomial multiplication reviewed in Section 2 along with binary field multiplication methods based on Toeplitz-matrix vector product (cf. [5], [8]). In Subsection 5.1, we compare the two-way split approaches, this includes results for the four-way split approach as the latter is an extension of the two-way approach. The three-way split approaches are compared in Subsection 5.2 followed by some concluding remarks.

5.1 Comparison of two-way split approaches

Table 8 summarizes the complexity of the two-way split approaches, i.e., non-recombined and recombined formulas, for the multiplication of polynomials.

TABLE 8

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

	Method	# AND	# XOR	Delay
TMVP	Fan-Hasan [5]	$n^{\log_2(3)}$	$5.5n^{\log_2(3)} - 6n + 0.5$	$D_\otimes + 2\log_2(n)D_\oplus$
	Recomb. Fan-Hasan [8]	$1.33n^{\log_2(3)}$	$5.17n^{\log_2(3)} - 7.5n + 1.5$	$D_\otimes + 2\log_2(n)D_\oplus$
Non-recomb. Polyn.	Two-way of [2], [14]	$n^{\log_2(3)}$	$5.5n^{\log_2(3)} - 7n + 1.5$	$D_\otimes + 3\log_2(n)D_\oplus$
	Two-way of [6]	$n^{\log_2(3)}$	$6n^{\log_2(3)} - 8n + 2$	$D_\otimes + 2\log_2(n)D_\oplus$
	Four-way formula of [2] (reviewed in Subsection 2.3)	$n^{\log_2(3)}$	$5.43n^{\log_2(3)} - 6.8n + 1.38$	$D_\otimes + 2.5\log_2(n)D_\oplus$
	Proposed four-way formula in Subsection 3.3	$n^{\log_2(3)}$	$5.88n^{\log_2(3)} - 8n + 2.13$	$D_\otimes + 2\log_2(n)D_\oplus$
Polyn Recomb.	Four-way recomb. of the four-way formula of [2]	$1.78n^{\log_2(3)}$	$4.56n^{\log_2(3)} - 8.9n + 3.63$	$D_\otimes + (2.5\log_2(n) - 2)D_\oplus$
	Four-way recomb. of the formula in Subsection 3.3	$1.78n^{\log_2(3)}$	$4.91n^{\log_2(3)} - 11n + 8.88$	$D_\otimes + (2\log_2(n) - 1)D_\oplus$

If we consider non-recombined polynomial multipliers in Table 8, we see that the four-way approach of [2] has a better space and time complexities compared to the two-way approach of [2], [14]. To the best of our knowledge this four-way approach has the smallest overall space complexity (AND and XOR gates combined) among all two or four-way formula. In terms of the time complexity, the two-way approach of [6] offers the best result among the existing non-recombined multipliers. Our proposed four-way formula achieves the same time complexity of [6] but with fewer gates. If we consider the space-time product complexity, the proposed four-way formula has the best result among all non-recombined two-way split formulas.

If we now compare all recombined and non-recombined polynomial approaches listed in Table 8, we see that, when we only consider the overall space complexity, the best approach is the four-way recombination of the four-way formula of [2], which requires $6.34n^{\log_2(3)} + O(n)$ gates operations. We also notice that the four-way recombined four-way split formula of Subsection 3.3 matches the best timing result of [6] while having a smaller space complexity.

If we now compare complexities of multipliers based on TMVP with multipliers based on polynomial formulas, we notice that the four-way recombination of four-way formula of [2] achieves the best results in terms of total number of gates. Specifically, the four-way recombination of the four-way formula of [2] requires $6.34n^{\log_2(3)}+O(n)$ gates while the recombined TMVP formula necessitates $6.5n^{\log_2(3)}+O(n)$. If we now compare the most efficient approach in terms of delay, we notice that the recombined TMVP formula remains the best approach: it requires $6.5n^{\log_2(3)} + O(n)$ gates and has a delay of $D_\otimes + 2\log_2(n)D_\oplus$ while the four-way recombination of the proposed four-way formula in Subsection 3.3 necessitates $6.69n^{\log_2(3)} + O(n)$ gates with a delay of $D_\otimes + 2\log_2(n)D_\oplus$. But, in design environments like ASIC the space of an XOR gate may be larger than AND gates [13]: in ASIC, an XOR gate are generally twice the size of an AND gate[1]. In this case, we obtain the following complexities:

- The recombination of TMVP formula of [8] has a space complexity of $11.66n^{\log_2(3)} + O(n)$ AND equivalents,
- The four-way recombination of the formula of Subsection 3.3 requires $11.60n^{\log_2(3)} + O(n)$ AND equivalents,

and this makes the proposed four-way recombination of the formula of Subsection 3.3 slightly better.

In Table 9, we have reported explicit complexities for three values of n of cryptographic interest. We then notice that:

- The total gate complexity of the proposed four-way formula in Subsection 3.3 improves the Fan *et al.* approach of [6] by $\sim 2\%$.
- The four-way recombination of the formula of Subsection 3.3 improves the total gate counts of Fan *et al.* approach [6] by $\sim 7\%$. And when we consider the ASIC environment where an XOR gate is twice the space requirement of an AND gate this improvement ratio goes up to $\sim 15\%$.
- The four-way recombination of four-way formula of [2] improves its non-recombined form by $\sim 4\%$ in terms of gate counts, and by $\sim 12\%$ in ASIC environment.
- The four-way recombination of the proposed four-way formula improves the gate count of the recombined TMVP multiplier by 0.8%, and by $\sim 2.7\%$ in ASIC environment.

These comparisons show that the recombination provides some significant gain in term of space complexity, and particularly in ASIC environment.

1. For a typical CMOS implementation, an XOR gate is made of twelve transistors while an AND gate uses only six

TABLE 9

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

Method		$n = 128$			$n = 256$			$n = 512$		
		#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay
TMVP	Fan-Hasan [5]	2187	11133	$D_\otimes + 14D_\oplus$	6561	34294	$D_\otimes + 16D_\oplus$	19683	104673	$D_\otimes + 18D_\oplus$
	Recomb. Fan-Hasan [8]	2916	10341	$D_\otimes + 14D_\oplus$	8748	31980	$D_\otimes + 16D_\oplus$	26244	97857	$D_\otimes + 18D_\oplus$
Non-recomb. Polyn.	Two-way of [2], [14]	2187	11134	$D_\otimes + 21D_\oplus$	6561	34295	$D_\otimes + 24D_\oplus$	19683	104674	$D_\otimes + 27D_\oplus$
	Two-way of [6]	2187	12100	$D_\otimes + 14D_\oplus$	6561	37320	$D_\otimes + 16D_\oplus$	19683	114004	$D_\otimes + 18D_\oplus$
	Four-way of [2] (reviewed in Subsection 2.3)	2187	11008	$D_\otimes + 17D_\oplus$	6561	33854	$D_\otimes + 20D_\oplus$	19683	103351	$D_\otimes + 22D_\oplus$
	Proposed four-way formula in Subsection 3.3	2187	11827	$D_\otimes + 14D_\oplus$	6561	36500	$D_\otimes + 16D_\oplus$	19683	111544	$D_\otimes + 18D_\oplus$
Recomb. Polyn.	Four-way recomb. with four-way form. of [2]	3888	8704	$D_\otimes + 15D_\oplus$	11664	27596	$D_\otimes + 18D_\oplus$	34992	84577	$D_\otimes + 20D_\oplus$
	Four-way recomb. of formula of Subsection 3.3	3888	9145	$D_\otimes + 14D_\oplus$	11664	29360	$D_\otimes + 16D_\oplus$	34992	90124	$D_\otimes + 18D_\oplus$

5.2 Comparison of three-way split approaches

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

TABLE 10

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

	Method	# AND	# XOR	Delay
TMVP	Fan-Hasan [5]	$n^{\log_3(6)}$	$4.8n^{\log_3(6)} - 5n + 0.2$	$D_\otimes + 3\log_3(n)D_\oplus$
	Recomb. Fan-Hasan [8]	$1.5n^{\log_2(3)}$	$4n^{\log_2(3)} - 6.33n + 1$	$D_\otimes + 3\log_3(n)D_\oplus$
Non-recomb.	Space efficient multiplier [4]	$n^{\log_3(6)}$	$5.27n^{\log_3(6)} - 6.67n + 1.4$	$D_\otimes + 4\log_3(n)D_\oplus$
	Delay efficient multiplier [6]	$n^{\log_3(6)}$	$5.33n^{\log_3(6)} - 7.33n + 2$	$D_\otimes + 3\log_3(n)D_\oplus$
	Proposed space efficient method in Subsection 3.1	$n^{\log_3(6)}$	$4.8n^{\log_3(6)} - 6n + 1.2$	$D_\otimes + 4\log_3(n)D_\oplus$
	Proposed delay efficient method in Subsection 3.1	$n^{\log_3(6)}$	$5.2n^{\log_3(6)} - 7n + 1.8$	$D_\otimes + 3\log_3(n)D_\oplus$
Recomb.	Recombined space efficient multiplier of Subsection 3.1	$1.5n^{\log_3(6)}$	$4n^{\log_3(6)} - 7.33n + 2$	$D_\otimes + (4\log_3(n) - 1)D_\oplus$
	Recombined space efficient multiplier of Subsection 3.2	$1.5n^{\log_3(6)}$	$4.33n^{\log_3(6)} - 9n + 5$	$D_\otimes + 3\log_3(n)D_\oplus$

Let us fist compare the non-recombined polynomial formulas. Based on the results in Table 10, we remark that the proposed method in Subsection 3.1 has the smallest space requirement. We also notice that the proposed delay efficient method in Subsection 3.1 matches the time complexity result of the

three-way split formula of [6] and has a smaller number of gates.

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

If we compare the polynomial formulas with the recombined TMVP three-way formula we notice that the latter compares favorably.

Indeed, the proposed block recombination of the space optimized three-way formula has, roughly, the same space complexity as the block recombined TMVP formula while having a larger delay. Similarly, the recombination on the delay efficient three-way formula reach the same delay as the recombined TMVP approach, but with a larger space complexity.

TABLE 11

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

	Method	$n = 243$			$n = 729$		
		# AND	# XOR	Delay	# AND	# XOR	Delay
TMVP	Fan-Hasan [5]	7776	36110	$D_\otimes + 15D_\oplus$	46656	220304	$D_\otimes + 18D_\oplus$
	Recomb. Fan-Hasan [8]	11664	29566	$D_\otimes + 15D_\oplus$	69984	182008	$D_\otimes + 18D_\oplus$
Non-recomb.	Space efficient multiplier [4]	7776	39335	$D_\otimes + 19D_\oplus$	46656	240863	$D_\otimes + 23D_\oplus$
	Delay efficient multiplier [6]	7776	39692	$D_\otimes + 15D_\oplus$	46656	243488	$D_\otimes + 18D_\oplus$
	Proposed space efficient method in Subsection 3.1	7776	35868	$D_\otimes + 19D_\oplus$	46656	219576	$D_\otimes + 23D_\oplus$
	Proposed delay efficient method in Subsection 3.1	7776	38736	$D_\otimes + 15D_\oplus$	46656	237510	$D_\otimes + 18D_\oplus$
Recomb.	Recombination of the space efficient method in Subsection 3.1	11664	29324	$D_\otimes + 19D_\oplus$	69984	181280	$D_\otimes + 23D_\oplus$
	Recombination of the delay efficient method in Subsection 3.2	11664	31514	$D_\otimes + 15D_\oplus$	69984	195620	$D_\otimes + 18D_\oplus$

These remarks are confirmed by the explicit complexities shown in Table 11 for the two cases $n = 243$ and $n = 729$. Indeed, the proposed space optimized three-way split formula improve the space requirement of [4] by $\sim 7\%$ while the proposed delay efficient formula improves the space requirement of Fan *et al.* three-way multiplier [6] by $\sim 2\%$. The block recombination enhances these optimizations since we obtain a reduction of the area requirement by 10 % for the space efficient formula and by 8% for the delay efficient formula.

Remark 4: We have compared both polynomial and TMVP approaches: indeed, both of them can be used to implement multipliers in finite field defined by an irreducible trinomial or pentanomial. But, we note that mention that, compared to Toeplitz matrix vector product, polynomial multiplication is found in more applications. For example, the subquadratic optimal normal basis multiplier presented in [3] requires polynomial multiplication only. Another example is the set of counter-measures preventing active and

passive side channel analysis on ECC proposed by Baek and Vasyltov in [1]: this approach is built upon binary polynomial arithmetic.

5.3 Concluding remarks

Multiplication of binary polynomials with sub-quadratic arithmetic complexity is often used in today's cryptographic systems, such as those based on elliptic curves and pairing [10], [7]. To this end, in this paper we have considered efficient bit parallel designs of sub-quadratic space complexity polynomial multipliers. Specifically, we have proposed improvements in terms of gate counts and delay to the existing best two- and three-way multipliers. The improvements have been achieved by identifying and removing some re-occurring computations. We have also proposed improvements to Bernstein's formula for the four-way split multiplication. More importantly, for the first time ever we have applied the block recombination approach to polynomial multiplication. This has provided a new area-time trade-offs and the least area-time product complexity among various multiplication schemes considered in the paper.

6 Aknowledgements

This work was supported in part by PAVOIS ANR 12 BS02 002 02.

References

[1] Y.-J. Baek and I. Vasyltov. How to Prevent DPA and Fault Attack in a Unified Way for ECC Scalar Multiplication - Ring Extension Method. In *Information Security Practice and Experience, Third International Conference, ISPEC 2007*, volume 4464 of *LNCS*, pages 225–237, 2007.

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

[3] D. J. Bernstein and T. Lange. Type-II Optimal Polynomial Bases. In *WAIFI 2010*, volume 6087 of *LNCS*, pages 41–61, 2010.

[4] 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.

[5] H. Fan and M.A. Hasan. A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields. *IEEE Trans. Computers*, 56(2):224–233, September 2007.

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

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

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

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

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

[11] 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.

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

[13] Taiwan Semiconductor Manufacturing Company Ltd. *TCBN65GPLUS - TSMC 65nm Core Library Databook*, June 2009.

[14] 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.



Murat Cenk received the B.S. degree in mathematics from Middle East Technical University, Ankara, Turkey in 2000, M.S. degree in mathematics and computer science from Cankaya University, Ankara, Turkey in 2003 and Ph.D. degree in cryptography from Institute of Applied Mathematics at Middle East Technical University in 2009. He has been a postdoctoral fellow in electrical and computer engineering at University of Waterloo, Canada since 2010. His research interests include algebraic complexity theory, cryptography, computer arithmetic and algebraic function fields over finite fields.



M. Anwar Hasan received the B.Sc. degree in electrical and electronic engineering, the M.Sc. degree in computer engineering, both from the Bangladesh University of Engineering and Technology, in 1986 and 1988, respectively, and the Ph.D. degree in electrical engineering from the University of Victoria in 1992. Dr. Hasan joined the Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada in 1993 and has been a full professor since 2002. At the University of Waterloo, he is also a member of the Centre for Applied Cryptographic Research, the Center for Wireless Communications, and the VLSI Research group. Dr. Hasan is a recipient of the Raihan Memorial Gold Medal. At the University of Victoria, he was awarded the President's Research Scholarship four times. At the University of Waterloo, he received a Faculty of Engineering Distinguished Performance Award in 2000, and Outstanding Performance Awards in 2004 and 2010. He served on the program and executive committees of several conferences. From 2000 to 2004, he was an associate editor of the IEEE Transactions on Computers. He is a licensed professional engineer of Ontario.



Christophe Negre received the MS degree in mathematics (2001) and the PhD degree in computer science (2004) from the Université Montpellier 2 (France). Since September 2004 , he has been an assistant professor with the DALI Team at the Université de Perpignan, France and LIRMM, CNRS and Université Montpellier 2. His research interests are in computer arithmetic and cryptography.