Block Recombination Approach for Subquadratic Space Complexity Binary Field Multiplication based on Toeplitz Matrix-Vector Product

M. A. Hasan[1], N. Méloni[1], A. H. Namin[1] and C. Negre[1,2]

[1]ECE Department and CACR, University of Waterloo, Waterloo, Ontario, Canada.

[2]Team DALI/ELIAUS, Université de Perpignan, Perpignan, France.

Abstract

In this paper, we present a new method for parallel binary finite field multiplication which results in subquadratic space complexity. The method is based on decomposing the building blocks of Fan-Hasan subquadratic Toeplitz matrix-vector multiplier. We reduce the space complexity of their architecture by recombining the building blocks. In comparison to other similar schemes available in the literature, our proposal presents a better space complexity while having the same time complexity. We also show that block recombination can be used for efficient implementation of the GHASH function of Galois Counter Mode (GCM).

Keywords. Binary field, subquadratic space complexity multiplier, Toeplitz matrix, block recombination.

1 Introduction

Finite fields have a wide range of applications in number theory, coding theory, and cryptography. Binary fields are specially attractive for high speed cryptographic applications since they are inherently suitable for hardware implementations. A binary field $\mathbb{F}_{2^n}$ is generally constructed as the set of binary polynomials modulo an irreducible polynomial P of degree n. An element of $\mathbb{F}_{2^n}$ is then a binary polynomial of degree smaller than n. Addition and multiplication in a binary field consist of polynomial addition and multiplication modulo P.

In general, field multipliers can be categorized into three different categories: bit-level, digit-level, and bit-parallel. Parallel multipliers present the fastest category of designs. Mainly, two types of parallel multipliers exist in the literature. The first type are the quadratic complexity architectures which require quadratic space complexity ($O(n^2)$ gates for their implementation), cf. [8], [12]. The second type are the subquadratic space complexity designs which require smaller number of gates for their implementation ($O(n^\delta), \delta < 2$), cf. [2], [11], [15], [1]. The latter presents practical architectures for hardware implementation of large field sizes, specially used in elliptic curve cryptographic applications [7], [10] ($163 < n < 571$).

There exist a limited number of algorithms to design subquadratic space complexity multipliers. The most well-known algorithm is based on the integer Karatsuba multiplication scheme, which has been widely applied to create subquadratic space complexity multipliers [6]. Architectures proposed in [2], [11] are based on this method. Another well-known algorithm is the Winograd short convolution algorithm [17] used for the same purpose in [15]. Chinese Reminder Theorem (CRT) is another example of such algorithms that results in subquadratic complexity multipliers [1].

Recently, Fan and Hasan proposed a new scheme to design subquadratic space complexity multipliers using Toeplitz matrix-vector product [3]. In their scheme they have used shifted polynomial basis for field elements representation to model the multiplication as a Toeplitz matrix product by a vector. Their method can also be used to design subquadratic space complexity polynomial, dual, weakly-dual and triangular basis multipliers. Then they proposed to perform two-way split or three-way split approaches to break down each matrix-vector product into a number of smaller size matrix-vector products. In [3], recursive use of this approach results in subquadratic space complexity multipliers which outperform the ones using Karatsuba, Winograd, and CRT.

In this paper, we propose an extension of the work of Fan and Hasan. We first decompose Fan-Hasan multiplier based on Toeplitz-matrix vector product (TMVP) into a number of different blocks. Each block performs independent computations (component matrix and vector formation, component multiplication and reconstruction). We then propose to recombine these blocks in order to reduce the space complexity. We first apply this recombination to a two-TMVPs-and-add architecture. We reorder the blocks and replace one reconstruction block by a smaller bitwise addition block. The space complexity is then reduced. After that, we use this recombination in a single TMVP multiplier. We express this product in terms of two-TMVPs-and-add. We obtain a multiplier which has less space and time complexity.

The rest of this work is organized as follows: in Section 2 we briefly review binary field multiplication and by recalling the multiplier of Fan and Hasan and we decompose it in four different blocks. In Section 3 we present our block recombination method in the special case of two-TMVPs-and-add architecture. Then we apply this method to reduce the space complexity of a single TMVP architecture and compare the results with previous similar approaches. We continue, in Section 4, by presenting an application of block recombination in the design of a space efficient architecture for GHASH computation. Finally Section 5 presents some concluding remarks.

2 Brief review of parallel binary field multiplication

Assume that $A = \sum_{i=0}^{n-1} a_i X^i$ and $B = \sum_{i=0}^{n-1} b_i X^i$ are two elements of $\mathbb{F}_{2^n}$, we can compute the product $C = A \times B \mod P$ as

$$\begin{array}{rcl} C & = & A \times B = \sum_{i=0}^{n-1} (AX^i) b_i \mod P \\ & = & \sum_{i=0}^{n-1} A^{(i)} b_i. \end{array}$$

Expanding the expression of B and considering that $A^{(i)} = AX^i \mod P$. This can be written through a matrix vector product

$$C = \left[\begin{array}{cccc} A^{(0)} & A^{(1)} & \dots & A^{(n-1)} \end{array} \right] \cdot B.$$

Direct hardware implementation of this matrix-vector product results in a quadratic area complexity circuit (i.e., it requires $O(n^2)$ gates). The two commonly used strategies to design an efficient hardware multiplier via the above matrix vector product are the following:

1) The choice of the polynomial P must provide an efficient computation of the columns $A^{(i)}$. Until now the all one polynomials (AOP) and trinomials seems to be the best possible choices [16], [5]. However, none of them exists for all degrees of n. Consequently other types of irreducible polynomials have been considered. Specifically, pentanomials of this form [13]

$$P = X^n + X^{k+2} + X^{k+1} + X^k + 1.$$

2) The second strategy consists of modifying the matrix

$$\left[\begin{array}{cccc} A^{(0)} & A^{(1)} & \dots & A^{(n-1)} \end{array} \right] \quad (1)$$

in order to obtain a Toeplitz matrix. Recall that an $n \times n$ Toeplitz matrix is a matrix $[t_{i,j}]_{0 \le i,j \le n-1}$ such that $t_{i,j} = t_{i-1,j-1}$ for $1 \le i,j \le n-1$. We will see in Subsection 2.1 that a Toeplitz matrix-vector product can be computed efficiently through a subquadratic complexity circuit (i.e., it requires $O(n^\delta)$ gates where $\delta < 2$). Generally we can obtain the Toeplitz form of the matrix in (1) by performing some row operations or column operations. In other words, we get this Toeplitz structure by using different bases of representation as it was shown by Hasan and Bhargava in [4]. This can be performed efficiently on fields defined by trinomials or specific pentanomials [3].

For the remainder of this paper we assume that binary field multiplication $C = A \times B$ has already been expressed as a Toeplitz matrix-vector product

$$C = T_A \cdot B.$$

2.1 Fan-Hasan subquadratic Toeplitz matrix-vector multiplier

In this section we recall the method used to build a subquadratic circuit which computes a Toeplitz matrix-vector product (TMVP) [3].

If $2|n$, Fan and Hasan proposed to use a *two-way split* approach shown in Table 1 to compute a matrix vector product $T \cdot V$, where T is an $n \times n$ Toeplitz matrix and V is a vector of size n. The two-way split approach breaks down a Toeplitz matrix-vector product of size n into three Toeplitz matrix-vector products of size $\frac{n}{2}$. If $3|n$, they proposed to use the *three-way split* approach which is also shown in Table 1. The three-way split approach results in six Toeplitz matrix-vector products of size $\frac{n}{3}$.

TABLE 1

Fan and Hasan two-way and three-way split formulas for TMVP [17], [3]

Two-way split	Three-way split
$T = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix}$	$T = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix}$
$T \cdot V = \begin{bmatrix} P_0 + P_1 \\ P_2 + P_1 \end{bmatrix}$ where $\begin{aligned} P_0 &= (T_0 + T_1) \cdot V_1, \\ P_1 &= T_1 \cdot (V_0 + V_1), \\ P_2 &= (T_1 + T_2) \cdot V_0, \end{aligned}$	$T \cdot V = \begin{bmatrix} P_0 + P_3 + P_4 \\ P_1 + P_3 + P_5 \\ P_2 + P_4 + P_5 \end{bmatrix}$ where $\begin{aligned} P_0 &= (T_0 + T_1 + T_2) \cdot V_2, \\ P_1 &= (T_1 + T_2 + T_3) \cdot V_1, \\ P_2 &= (T_2 + T_3 + T_4) \cdot V_0, \\ P_3 &= T_1 \cdot (V_1 + V_2), \\ P_4 &= T_2 \cdot (V_0 + V_2), \\ P_5 &= T_3 \cdot (V_0 + V_1), \end{aligned}$

If n is a power of 2 or a power of 3, Fan and Hasan also proposed to recursively use the formulas given in Table 1 to perform $T \cdot V$. Using this recursive process through parallel computation, the resulting multiplier would have the complexities given in Table 2 (cf. [3]). In this table D_A represents the delay of a two-input AND gate and D_X the delay of a two-input XOR gate. It is also possible to design subquadratic TMVP multipliers for size $n = 3^i 2^j$ by combining the two-way and three-way split approaches in the recursive computations.

TABLE 2

Asymptotic complexities of a single Fan-Hasan TMVP

	Two-way split method	Three-way split method
# AND	$n^{\log_2(3)}$	$n^{\log_3(6)}$
# XOR	$5.5n^{\log_2(3)} - 6n + 0.5$	$\frac{24}{5}n^{\log_3(6)} - 5n + \frac{1}{5}$
Delay	$D_A + 2\log_2(n)D_X$	$D_A + 3\log_3(n)D_X$

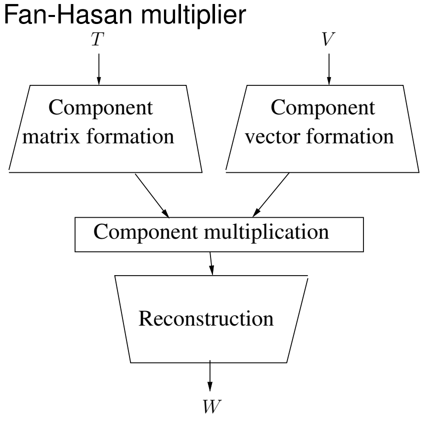
2.2 Block decomposition of Fan-Hasan multiplier

In this subsection we decompose the Fan-Hasan multiplier into a number of independent blocks. We will then evaluate the complexity of each block. We decompose the recursive formulas of Table 1 in four independent computations

- *Component matrix formation (CMF).* We call component matrix formation the recursive matrix computation of Table 1. For example, for the two-way split approach, the first recursion on T computes $T_0 + T_1, T_1, T_1 + T_2$. This means that the component matrix formation can be expressed as $CMF(T) = (CMF(T_0+T_1), CMF(T_1), CMF(T_1+T_2))$. In the sequel we will often refer to $CMF(T)$ as the *component representation* of the Toeplitz matrix T.
- *Component vector formation (CVF).* This corresponds to the recursive computation on the vector V in Table 1. For example, for the two-way split approach, we see that recursion is applied to V_1, V_0+V_1, V_0. Consequently the component vector formation is expressed as $CVF(V) = (CFV(V_1), CVF(V_0 + V_1), CVF(V_0))$. In the sequel we will often refer to $CVF(V)$ as the *component representation* of the vector V.
- *Component multiplication (CM).* In Table 1 we see that $CMF(T)$ and $CVF(V)$ are multiplied component by component at the end of the recursion (this corresponds to the recursive multiplication yielding P_0, P_1, P_2).
- *Reconstruction (R).* The last operation is the reconstruction of the product $W = T \cdot V$ from the component multiplication of $CTF(T)$ and $CVF(V)$. For example, for the two-way split case, let $\hat{W}$ be equal to the component multiplication of V and T. If we split $\hat{W} = [\hat{W}_0, \hat{W}_1, \hat{W}_2]$, then the formula in Table 1 states that $W = R(\hat{W}) = (R(\hat{W}_0) + R(\hat{W}_1), R(\hat{W}_1) + R(\hat{W}_2)$.

Then we can split the Fan-Hasan multiplier in four distinct blocks (CMF,CVF,CM and R) as shown in Fig. 1.

Fig. 1. Block decomposition of the Fan-Hasan multiplier



Example 1. Here we develop a detailed logical diagram of the hardware of the Fan-Hasan multiplier for $n = 4$, which computes the following matrix-vector product

$$\begin{bmatrix} t_3 & t_2 & t_1 & t_0 \\ t_4 & t_3 & t_2 & t_1 \\ t_5 & t_4 & t_3 & t_2 \\ t_6 & t_5 & t_4 & t_3 \end{bmatrix} \cdot \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

For the CMF, the formula of Table 1 for the first recursion gives

$$T_0 + T_1 = \begin{bmatrix} t_3 & t_2 \\ t_4 & t_3 \end{bmatrix} + \begin{bmatrix} t_1 & t_0 \\ t_2 & t_1 \end{bmatrix}, \quad T_1, \quad T_1 + T_2 = \begin{bmatrix} t_5 & t_4 \\ t_6 & t_5 \end{bmatrix} + \begin{bmatrix} t_3 & t_2 \\ t_4 & t_3 \end{bmatrix}.$$

This means that in the first recursion we calculate the following $(t_3 + t_1, t_2 + t_0, t_4 + t_2), (t_2, t_3, t_4), (t_5 + t_3, t_2 + t_4, t_6 + t_4)$. Then the formula of Table 1 are applied to the three 2×2 matrices. This results in circuit depicted in CMF block in Fig. 2. The same is done for the CVF. Indeed if we apply the formula of Table 1 to V we get

$$\begin{bmatrix} v_2 \\ v_3 \end{bmatrix}, \begin{bmatrix} v_0 \\ v_1 \end{bmatrix} + \begin{bmatrix} v_2 \\ v_3 \end{bmatrix}, \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}.$$

Then the formulas are applied to each of the three resulting vectors. We then obtain the CVF block shown in Fig. 2. The component multiplication is then just a level of parallel AND operations with inputs $CMF(T)$ and $CVF(V)$. Then for the reconstruction we apply formulas of Table 1 at each group of three bits of the nine bits output by CM. Each group gives a vector of length two, and the reconstruction is then applied to these three vectors of size two. The resulting block decomposion of the Fan-Hasan multiplier is given in Fig. 2

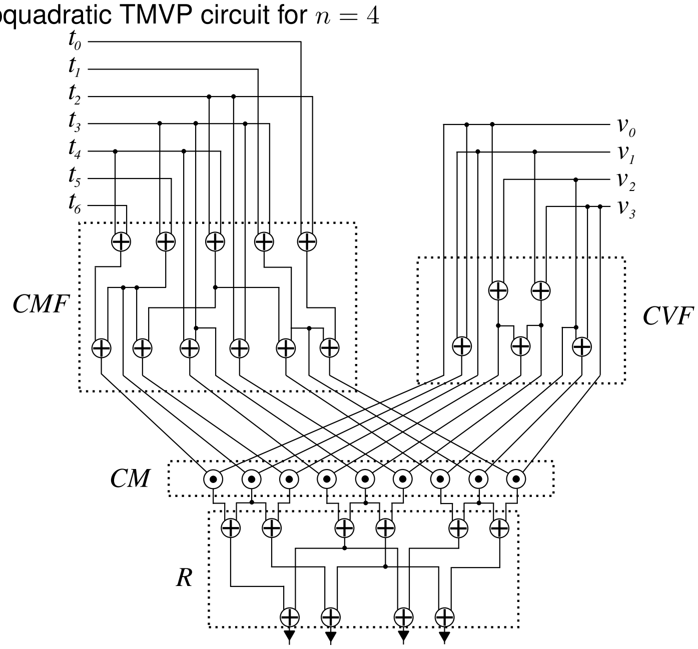
2.3 Block complexity in Fan-Hasan multiplier

We can evaluate the complexity of the Fan-Hasan architecture by counting separately the contribution in the space ($\mathcal{S}$) and delay of time ($\mathcal{D}$) complexities of above mentioned four blocks. The complexities are named as follows

- $\mathcal{S}^T_{2,\oplus}, \mathcal{D}^T_2$ for the component matrix formation,
- $\mathcal{S}^V_{2,\oplus}, \mathcal{D}^V_2$ for the component vector formation,
- $\mathcal{S}^M_{2,\oplus}, \mathcal{D}^M_2$ for the component multiplication,
- $\mathcal{S}^R_{2,\oplus}, \mathcal{D}^R_2$ for the vector reconstruction.

We evaluate in detail the complexities $\mathcal{S}^T_{2,\oplus}$ and $\mathcal{D}^T_2$ for the two-way split approach. The other block complexities can be evaluated in the same way. Using the formulas of Table 1, we would like to express $\mathcal{S}^T_{2,\oplus}(n), \mathcal{D}^T_2(n)$ in terms of $\mathcal{S}^T_{2,\oplus}(n/2)$ and $\mathcal{D}^T_2(n/2)$ respectively. We need to evaluate the cost of the computation of T_0+T_1 and T_1+T_2. A direct method to compute the matrix additions T_0+T_1 and T_1+T_2 needs $2(n-1)$ additions. As mentioned in [3], some terms can be reused in the computation of $T_0 + T_1$

Fig. 2. Fan-Hasan Subquadratic TMVP circuit for $n = 4$



and $T_1 + T_2$. Specifically, in the computation of $T_1 + T_2$ we only need to compute the first column of $T_1 + T_2$. Indeed the first row of $T_1 + T_2$ also appears in the first column of $T_0 + T_1$. Therefore, the total number of XOR gates required for the two matrix additions is equal to $\frac{3n}{2} - 1$. We obtain the following inductive relations for $\mathcal{S}^T_{2,\oplus}$ and $\mathcal{D}^T_2$.

$$\left\{\begin{array}{rcl} \mathcal{S}^T_{2,\oplus}(1) & = & 0, \\ \mathcal{S}^T_{2,\oplus}(n) & = & 3\mathcal{S}^T_{2,\oplus}(n/2) + 3n/2 - 1, \end{array}\right. \qquad \left\{\begin{array}{rcl} \mathcal{D}^T_2(1) & = & 0, \\ \mathcal{D}^T_2(n) & = & \mathcal{D}^T_2(n/2) + D_X, \end{array}\right. \tag{2}$$

The following lemma of [3] helps us to get the non-inductive expression of $\mathcal{S}^T_{2,\oplus}$ and $\mathcal{D}^T_2$ in terms of n.

Lemma 1. *Let a, b and i be positive integers. Let $n = b^i$, $a \neq b$ and $a \neq 1$. The solution of the inductive relation* $\left\{\begin{array}{rcl} r_1 & = & e, \\ r_n & = & ar_{n/b} + cn + d, \end{array}\right.$ *is*

$$r_n = en^{\log_b(a)} + \frac{bc(n^{\log_b(a)} - n)}{a - b} + \frac{d(n^{\log_b(a)} - 1)}{a - 1}.$$

Applying Lemma 1 to the inductive relation to (2), we obtain the following

$$\left\{\begin{array}{rcl} \mathcal{S}^T_{2,\oplus}(n) & = & 2.5n^{\log_2(3)} - 3n + 0.5, \\ \mathcal{D}^T_{2,\oplus}(n) & = & \log_2(n) D_X. \end{array}\right.$$

We can use the same method for both two-way split and three-way split approaches for the component vector formation of V, the component multiplication and the vector reconstruction. The space and time complexities are summarized in Table 3.

TABLE 3

Space and time complexities of different blocks in the Fan-Hasan multiplier

Computation	Split	Recursive formulas		Complexities
Component matrix formation	2	$\mathcal{S}^T_{2,\oplus}(n) = \frac{3n}{2} - 1 + 3\mathcal{S}^T_{2,\oplus}(\frac{n}{2})$ $\mathcal{D}^T(n) = D_X + \mathcal{D}^T(n/2)$	$\mathcal{S}^T_{2,\oplus}(1) = 0$ $\mathcal{D}^T(1) = 0$	$\mathcal{S}^T_{2,\oplus}(n) = \frac{5}{2}n^{\log_2(3)} - 3n + \frac{1}{2}$ $\mathcal{D}^T_2(n) = \log_2(n)D_X$
Component vector formation	2	$\mathcal{S}^V_{2,\oplus}(n) = \frac{n}{2} + 3\mathcal{S}^V_{2,\oplus}(\frac{n}{2})$ $\mathcal{D}^V(n) = D_X + \mathcal{D}^V(n/2)$	$\mathcal{S}^V_{2,\oplus}(1) = 0$ $\mathcal{D}^V(1) = 0$	$\mathcal{S}^V_{2,\oplus}(n) = n^{\log_2(3)} - n$ $\mathcal{D}^V(n) = \log_2(n)D_X$
Component multiplication	2	$\mathcal{S}^M_{2,\otimes}(n) = 3\mathcal{S}^M_{2,\otimes}(\frac{n}{2})$ $\mathcal{D}^M(n) = \mathcal{D}^M(n/2)$	$\mathcal{S}^M_{2,\otimes}(1) = 1$ $\mathcal{D}^M(1) = D_A$	$\mathcal{S}^M_{2,\otimes}(n) = n^{\log_2(3)}$ $\mathcal{D}^M(n) = D_A$
Reconstruction	2	$\mathcal{S}^R_{2,\oplus}(n) = n + 3\mathcal{S}^R_{2,\oplus}(\frac{n}{2})$ $\mathcal{D}^R(n) = D_X + \mathcal{D}^R(n/2)$	$\mathcal{S}^R_{2,\oplus}(1) = 0$ $\mathcal{D}^R(1) = 0$	$\mathcal{S}^R_{2,\oplus}(n) = 2n^{\log_2(3)} - 2n$ $\mathcal{D}^R(n) = \log_2(n)D_X$
Component matrix formation	3	$\mathcal{S}^T_{3,\oplus}(n) = 2n - 1 + 6\mathcal{S}^T_{3,\oplus}(\frac{n}{3})$ $\mathcal{D}^T(n) = 2D_X + \mathcal{D}^T(n/3)$	$\mathcal{S}^T_{3,\oplus}(1) = 0$ $\mathcal{D}^T(1) = 0$	$\mathcal{S}^T_{3,\oplus}(n) = \frac{9}{5}n^{\log_3(6)} - 2n + \frac{1}{5}$ $\mathcal{D}^T(n) = 2\log_3(n)D_X$
Component vector formation	3	$\mathcal{S}^V_{3,\oplus}(n) = n + 6\mathcal{S}^V_{3,\oplus}(\frac{n}{3})$ $\mathcal{D}^V(n) = D_X + \mathcal{D}^V(n/3)$	$\mathcal{S}^V_{3,\oplus}(1) = 0$ $\mathcal{D}^V(1) = 0$	$\mathcal{S}^V_{3,\oplus}(n) = n^{\log_3(6)} - n$ $\mathcal{D}^V(n) = \log_3(n)D_X$
Component multiplication	3	$\mathcal{S}^M_{3,\otimes}(n) = 6\mathcal{S}^M_{3,\otimes}(\frac{n}{3})$ $\mathcal{D}^M(n) = \mathcal{D}^T(n/3)$	$\mathcal{S}^M_{3,\otimes}(1) = 1$ $\mathcal{D}^M(1) = 0$	$\mathcal{S}^M_{3,\otimes}(n) = n^{\log_3(6)}$ $\mathcal{D}^M(n) = D_A$
Reconstruction	3	$\mathcal{S}^R_{3,\oplus}(n) = 2n + 6\mathcal{S}^R_{3,\oplus}(\frac{n}{3})$ $\mathcal{D}^R(n) = D_X + \mathcal{D}^R(n/3)$	$\mathcal{S}^R_{3,\oplus}(1) = 0$ $\mathcal{D}^R(1) = 0$	$\mathcal{S}^R_{3,\oplus}(n) = 2n^{\log_3(6)} - 2n$ $\mathcal{D}^R(n) = 2\log_3(n)D_X$

Remark 1. The component representation $CVF(V)$ of the vector V and $CMF(T)$ of the Toeplitz matrix T have the same number of bits. These bits are multiplied through parallel AND gates in component multiplication. A consequence is that the number of bits in component representation is equal to the number of AND gates inside the CM block. It means that in the two-way split approach the number of bits of the component representation of either V and T is equal to $n^{\log_2(3)}$. In three-way split approach the number of bits of the component representation of either V and T is equal to $n^{\log_3(6)}$.

3 Block recombination of Fan-Hasan multiplier

We would like to use the block decomposition of the Fan-Hasan multiplier in order to reduce its space complexity by recombining the underlying blocks. We modify only the first step in the recursive computation of the Fan-Hasan multiplier. For the two-way split approach, we split T in 4 blocks and V in 2 blocks and we perform $T \cdot V$ using direct computation of a matrix vector product shown in (3).

$$\begin{bmatrix} W_0 \\ W_1 \end{bmatrix} = \begin{bmatrix} T_0 & T_1 \\ T_2 & T_0 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} T_0 \cdot V_0 + T_1 \cdot V_1 \\ T_2 \cdot V_0 + T_0 \cdot V_1 \end{bmatrix}. \tag{3}$$

We note that the Toeplitz matrix-vector product is reduced to the computation of two instances of independent *two-TMVPs-and-add*, i.e., $T_0 \cdot V_0 + T_1 \cdot V_1$ and $T_2 \cdot V_0 + T_0 \cdot V_1$.

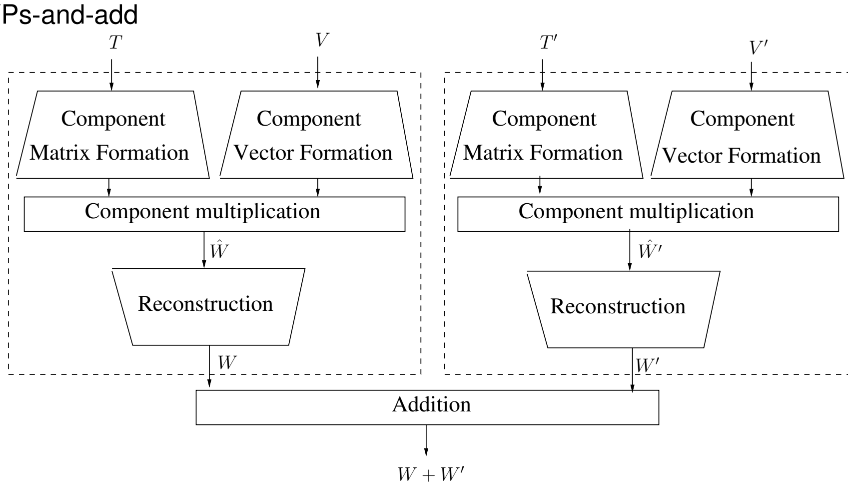
3.1 Recombination of two parallel TMVPs and add

We consider here the problem to compute two parallel TMVPs followed by an addition. Specifically, if T and T' are $n \times n$ Toeplitz matrices, and V and V' two vectors of size n, we want to compute

$$W = T \cdot V + T' \cdot V'.$$

A simple way to design an architecture to perform this operation consists of two parallel subquadratic TMVP multipliers and n XOR gates to add the two products. This architecture is shown in Fig. 3.

Fig. 3. two-TMVPs-and-add



In order to reduce the space complexity, we try to combine parts of the two multipliers. This can be achieved by joining the reconstruction part of the multipliers using the following property of vector reconstruction.

Property 1. *Let $\hat{W}$ and $\hat{W}'$ be two vectors of size $n^{\log_2(3)}$ (resp. $n^{\log_3(6)}$ for the three-way split case). Let R denote the reconstruction function. Then we have*

$$R(\hat{W}) + R(\hat{W}') = R(\hat{W} + \hat{W}'). \tag{4}$$

Proof: We first prove it by induction for the two-way split case. When $n = 1$, we have $U = R(\hat{U}) = \hat{U}$ for all U which implies that $R(\hat{W}) + R(\hat{W}') = \hat{W} + \hat{W}' = R(\hat{W} + \hat{W}')$. Let us now show it for $n = 2^s$ assuming that (4) is true for $n = 2^{s-1}$. We decompose $\hat{W}$ and $\hat{W}'$ in three part of size $\frac{n^{\log_2(3)}}{3}$

$$\begin{aligned} \hat{W} &= [\hat{W}_0, \hat{W}_1, \hat{W}_2], \\ \hat{W}' &= [\hat{W}'_0, \hat{W}'_1, \hat{W}'_2]. \end{aligned}$$

We then apply the formula defining R given in Table 1 to these expressions of $\hat{W}$ and $\hat{W}'$

$$\begin{aligned} R(\hat{W}) &= [R(\hat{W}_0) + R(\hat{W}_1), R(\hat{W}_2) + R(\hat{W}_1)], \\ R(\hat{W}') &= [R(\hat{W}'_0) + R(\hat{W}'_1), R(\hat{W}'_2) + R(\hat{W}'_1)]. \end{aligned}$$

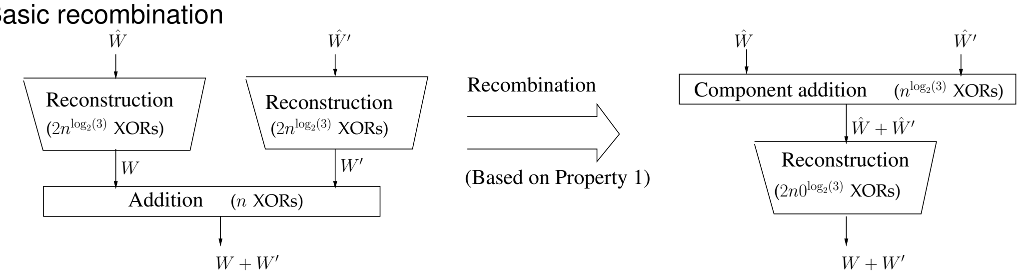
Now we add these two identities and then we apply the induction hypothesis, we obtain that

$$\begin{aligned} R(\hat{W}) + R(\hat{W}') &= [R(\hat{W}_0) + R(\hat{W}'_0) + R(\hat{W}_1) + R(\hat{W}'_1), R(\hat{W}_2) + R(\hat{W}'_2) + R(\hat{W}_1) + R(\hat{W}'_1)] \\ &= [R(\hat{W}_0 + \hat{W}'_0) + R(\hat{W}_1 + \hat{W}'_1), R(\hat{W}_2 + \hat{W}'_2) + R(\hat{W}_1 + \hat{W}'_1)] \\ &= R(\hat{W} + \hat{W}'). \end{aligned}$$

This ends the proof for the two-way split case. We give the proof of the three-way split case in the appendix. □

We apply the previous property to reorganize the block decomposition of two-TMVPs-and-add architecture. For the sake of simplicity we consider only the case of two-way split multipliers. Similar recombination could be performed on the three-way split multiplier. Property 1 allows us to perform the block recombination depicted in Fig. 4 for the two-way split case. In Fig. 4, the structure on the left side is functionally equal to the new structure on the right side.

Fig. 4. Basic recombination



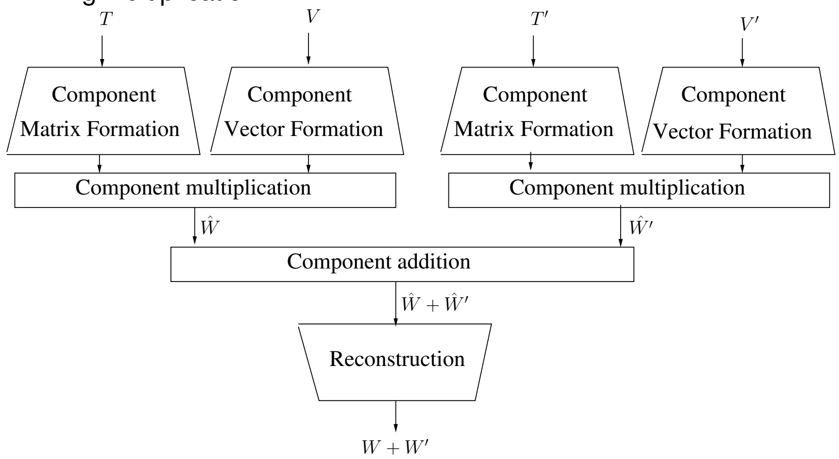
Reconstruction blocks are relatively space consuming since they require $(2n^{\log_3(2)} - 2n)$ XOR gates each. The recombination of Fig. 4 reduces the number of reconstruction blocks from two to one. At the same time the number of XOR gates in the component addition block becomes $n^{\log_3(2)}$ since the size of component representation is equal to $n^{\log_2(3)}$ bits. The space complexity of the architecture in the left part of Fig. 4 is equal to $4n^{\log_2(3)} - n$ XOR gates. The space complexity for the recombined architecture in the right part of Fig. 4 is equal to $3n^{\log_2(3)} - 2n$ XOR gates. Consequently this recombination reduces the space complexity by $(n^{\log_3(2)} + n)$ XOR gates.

Now, we include this recombination scheme in the two-TMVPs-and-add architecture. The resulting recombined architecture for two-TMVPs-and-add in given in Fig. 5.

Complexity. We evaluate the complexity of the architecture depicted in Fig. 5. In the new architecture there exists two CTFs, two CVFs, two CMs, one CAs and one reconstruction block. If we denote $S_{\oplus}^{A}(n)$ the number of XORs gates in the component addition, we obtain the following expression for the number of AND gates and XOR gates of the architecture in Fig. 5

$$\begin{aligned} S_{\oplus}(n) &= 2S_{\oplus}^{T}(n) + 2S_{\oplus}^{V}(n) + 2S_{\oplus}^{A}(n) + S_{\oplus}^{R}(n), \\ S_{\otimes}(n) &= S_{\otimes}^{M}(n). \end{aligned}$$

Fig. 5. Scheme of mixing multiplication



Using the complexity of each block given in Table 3, we obtain the complexities listed in Table 4. In this Table we also give the complexities of the architecture in Fig. 3 for comparison. We can see that, for the two-way split case, the recombination performed in Fig. 5 reduces the number of XOR gates by $(n^{\log_2(3)} + n)$ having the same delay. The space and delay complexities for the three-way split approach can be calculated in the same way, the results are also summarized in Table 4.

TABLE 4

Complexities of two-TMVPs-and-add architectures

Arch.	Split size	Space		Delay	
		#XOR	#AND	$\#D_X$	$\#D_A$
Fig. 3	2	$11n^{\log_2(3)} - 11n + 1$	$2n^{\log_2(3)}$	$2\log_2(n) + 1$	1
Fig. 5	2	$10n^{\log_2(3)} - 10n + 1$	$2n^{\log_2(3)}$	$2\log_2(n) + 1$	1
Fig. 3	3	$\frac{48}{5}n^{\log_3(6)} - 9n + 2/5$	$2n^{\log_3(6)}$	$3\log_3(n) + 1$	1
Fig. 5	3	$\frac{43}{5}n^{\log_3(6)} - 8n + 2/5$	$2n^{\log_3(6)}$	$3\log_3(n) + 1$	1

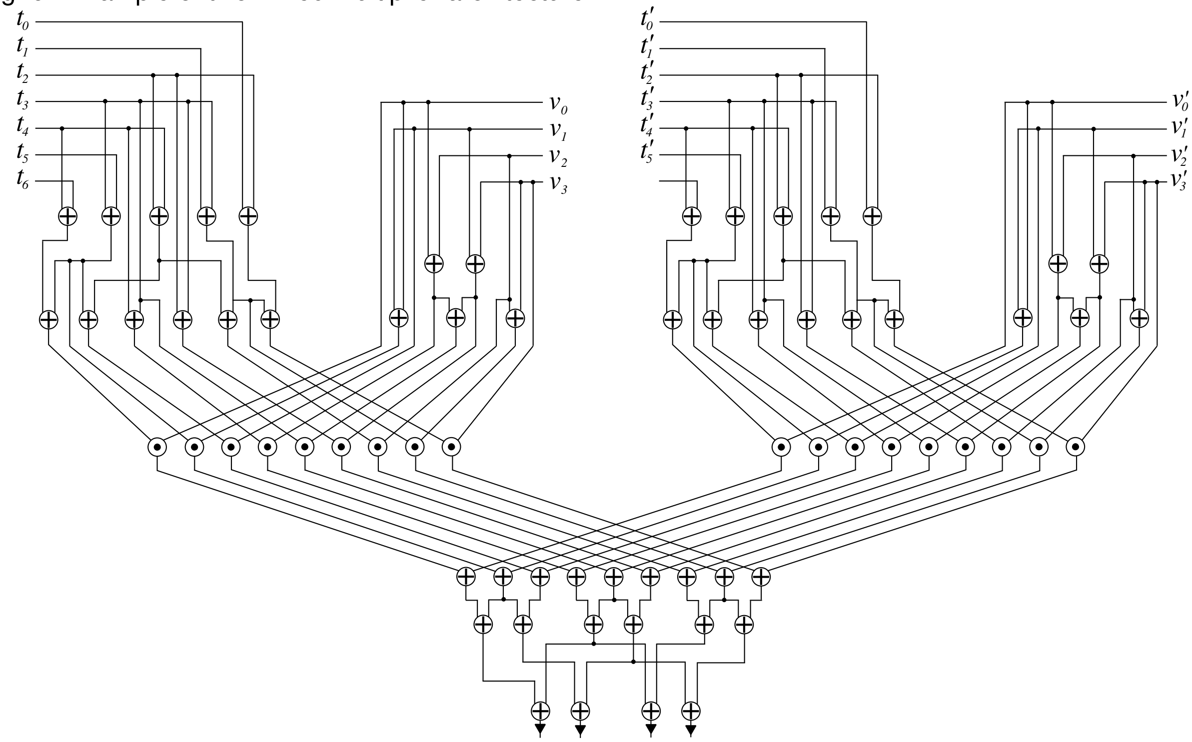
Example 2. In Fig. 6, we give the recombined hardware architecture which computes $C = T \cdot V + T' \cdot V'$ where Toeplitz matrices T and T' and vectors V and V' have size $n = 4$. $\diamond$

The modification on the two-TMVPs-and-add architecture can be formulated in an algorithmic format as presented in Algorithm 1 below for the two-way split approach. The algorithm for the three-way split approach can be obtained using the same methodology.

The following theorem proves the correctness of Algorithm 1.

Theorem 1. *Let T and T' be two $n \times n$ Toeplitz matrices and V and V' be two vectors of length n with entries*

Fig. 6. Example of two Mixed multiplier architecture



in $\mathbb{F}_2$ *and* $n = 2^s$*. Then Algorithm 1 returns the correct value of* $T \cdot V + T' \cdot V'$.

Proof: This can be proved by induction on $s = \log_2(n)$. For $s = 0$, i.e., $n = 1$, Algorithm 1 returns $T \cdot V + T' \cdot V'$ which is correct.

Now we assume that Algorithm 1 returns the correct value for $n = 2^i$ where $i = 0, 1, 2, \ldots, s$ and we show that under this assumption it returns the correct value for $n = 2^{s+1}$ as well. For this case the algorithm recursively computes

$$\begin{aligned} P_0 &\leftarrow \text{Two-TMVPs-and-Add}(T_0 + T_1, V_1, T'_0 + T'_1, V'_1), \\ P_1 &\leftarrow \text{Two-TMVPs-and-Add}((T_1 + T_2), V_0, (T'_1 + T'_2), V'_0), \\ P_2 &\leftarrow \text{Two-TMVPs-and-Add}(T_1, (V_0 + V_1), T'_1, (V'_0 + V'_1)). \end{aligned}$$

By induction, since matrices and vectors have size 2^s, the recursive calls to Algorithm 1 return the correct values. This means that

$$\begin{aligned} P_0 &= (T_0 + T_1) \cdot V_1 + (T'_0 + T'_1) \cdot V'_1, \\ P_1 &= (T_1 + T_2) \cdot V_0 + (T'_1 + T'_2) \cdot V'_0, \\ P_2 &= T_1 \cdot (V_0 + V_1) + T'_1 \cdot (V'_0 + V'_1). \end{aligned}$$

Then, the final returned vector satisfies

Algorithm 1 Two-TMVPs-and-Add(T_1, V_1, T_2, V_2)

Require: T, V, T', V' where T and T' are $n \times n$ Toeplitz matrices and V and V' are vectors of length $n = 2^s$ each.

Ensure: $T \cdot V + T' \cdot V'$.

if $n = 1$ **then**

Return($T \cdot V + T' \cdot V'$)

else

Split $T \cdot V = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix}, \quad T' \cdot V' = \begin{bmatrix} T_1' & T_0' \\ T_2' & T_1' \end{bmatrix} \cdot \begin{bmatrix} V_0' \\ V_1' \end{bmatrix}$

$P_0 =$ Two-TMVPs-and-Add$(T_0 + T_1, V_1, T_0' + T_1', V_1')$

$P_1 =$ Two-TMVPs-and-Add$((T_1 + T_2), V_0, (T_1' + T_2'), V_0')$

$P_2 =$ Two-TMVPs-and-Add$(T_1, (V_0 + V_1), T_1', (V_0' + V_1'))$

Return($\begin{bmatrix} P_0 + P_2 \\ P_1 + P_2 \end{bmatrix}$)

end if

$$\begin{aligned}
\begin{bmatrix} P_0 + P_2 \\ P_1 + P_2 \end{bmatrix} &= \begin{bmatrix} (T_0 + T_1) \cdot V_1 + (T_0' + T_1') \cdot V_1' + (T_1 + T_2) \cdot V_0 + (T_1' + T_2') \cdot V_0' \\ (T_1 + T_2) \cdot V_0 + (T_1' + T_2') \cdot V_0' + T_1 \cdot (V_0 + V_1) + T_1' \cdot (V_0' + V_1') \end{bmatrix} \\
&= \begin{bmatrix} (T_0 + T_1) \cdot V_1 + (T_1 + T_2) \cdot V_0 \\ (T_1 + T_2) \cdot V_0 + T_1 \cdot (V_0 + V_1) \end{bmatrix} + \begin{bmatrix} (T_0' + T_1') \cdot V_1' + (T_1' + T_2') \cdot V_0' \\ (T_1' + T_2') \cdot V_0' + T_1' \cdot (V_0' + V_1') \end{bmatrix} \\
&= T \cdot V + T' \cdot V'
\end{aligned}$$

□

Remark 2.
- The problem of computing of two-TMVPs-and-add could be seen as a single matrix vector product. Indeed, if we consider an $n \times 2n$ matrix T which can be split into $n \times n$ Toeplitz sub-matrices T_1 and T_2 as follows
$$T = \begin{bmatrix} T_1 & T_2 \end{bmatrix}.$$
Let $V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$ be a vector of size $2n$. Then the matrix-vector product $T \cdot V$ can be computed as $T \cdot V = T_1 \cdot V_1 + T_2 \cdot V_2$. This is a two-TMVPs-and-add computation. We can thus compute it by applying the method previously presented in this section.
- The recombination of Fig. 5 and its corresponding Algorithm 1 formulation can be generalized to k-TMVPs-and-add with $k \geq 2$. This extends directly from the results on the two-TMVPs-and-add case.

3.2 Recombination for a single two-way split TMVP multiplier

We consider now the problem to compute one single Toeplitz matrix-vector product. Specifically, let T be an $n \times n$ matrix and V be a vector of size n. We would like to compute efficiently

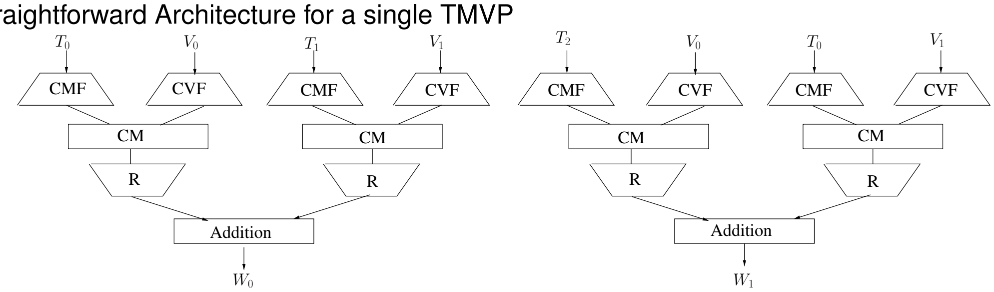
$$W = T \cdot V.$$

Our strategy is to modify the first recursion of the Fan-Hasan multiplier, in order to use reduce the space complexity by block recombination. We split T in 4 blocks and V in 2 blocks and we perform $T \cdot V$ through a direct computation. We obtain

$$\begin{bmatrix} W_0 \\ W_1 \end{bmatrix} = \begin{bmatrix} T_0 & T_1 \\ T_2 & T_0 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} T_0 \cdot V_0 + T_1 \cdot V_1 \\ T_2 \cdot V_0 + T_0 \cdot V_1 \end{bmatrix} \tag{5}$$

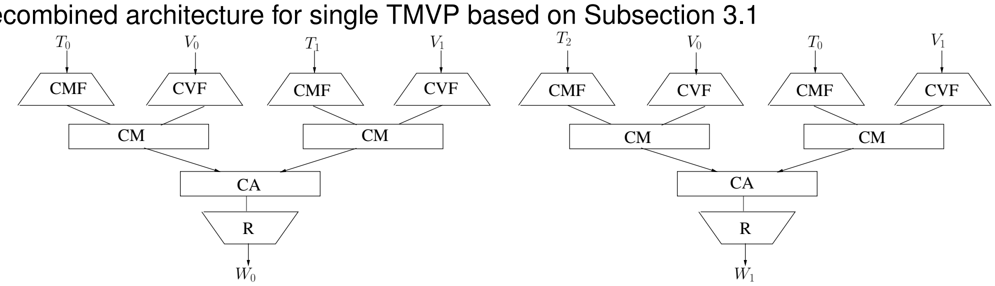
Now if we perform the four TMVPs of size $n/2$ using the the Fan-Hasan multiplier we obtain the architecture shown in Fig. 7.

Fig. 7. Straightforward Architecture for a single TMVP



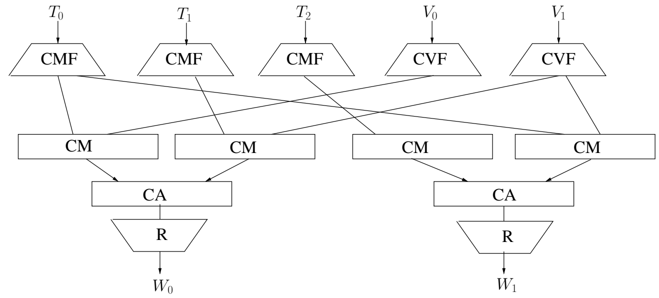
Now if we use the recombination of Subsection 3.1 we obtain the architecture of Fig. 8.

Fig. 8. Recombined architecture for single TMVP based on Subsection 3.1



We remark that several blocks are performing exactly the same computation. Specifically, there are two blocks which perform the component matrix formation of T_0, two blocks perform the component vector formation of V_0, and two blocks perform the component vector formation of V_1. We can thus keep only one of each of the above mentioned blocks. After doing this we obtain the architecture depicted in Fig. 9.

Fig. 9. Simplified single TMVP Architecture



Let us now evaluate the complexity of the architecture of Fig. 9. The space complexity of this approach is expressed in terms of the space complexity of the different blocks: $\mathcal{S}_{2,\oplus}^T$ for component Toeplitz matrix formation, $\mathcal{S}_{2,\oplus}^T$ for component vector formation, $\mathcal{S}_{2,\oplus}^R$ for reconstruction, $\mathcal{S}_{2,\otimes}^M$ for component multiply and $\mathcal{S}_{2,\oplus}^A$ component addition. Then we have

$$\mathcal{S}_{2,\oplus} = 3\mathcal{S}_{2,\oplus}^T(\frac{n}{2}) + 2\mathcal{S}_{2,\oplus}^V(\frac{n}{2}) + 2\mathcal{S}_{2,\oplus}^R(\frac{n}{2}) + 2\mathcal{S}_{2,\oplus}^A(\frac{n}{2})$$

Using Table 3 and Remark 1

$$\mathcal{S}_{\oplus}(n) = \frac{3}{2} + \frac{31}{6}n^{\log_2(3)} - \frac{15}{2},$$

and for the number of AND gates we have

$$\mathcal{S}_{\otimes}(n) = 4\mathcal{S}_{2,\otimes}^M(\frac{n}{2}) = \frac{4}{3}n^{\log_2(3)}.$$

The time complexity is equal to $D_A + 2\log_2(n)D_X$.

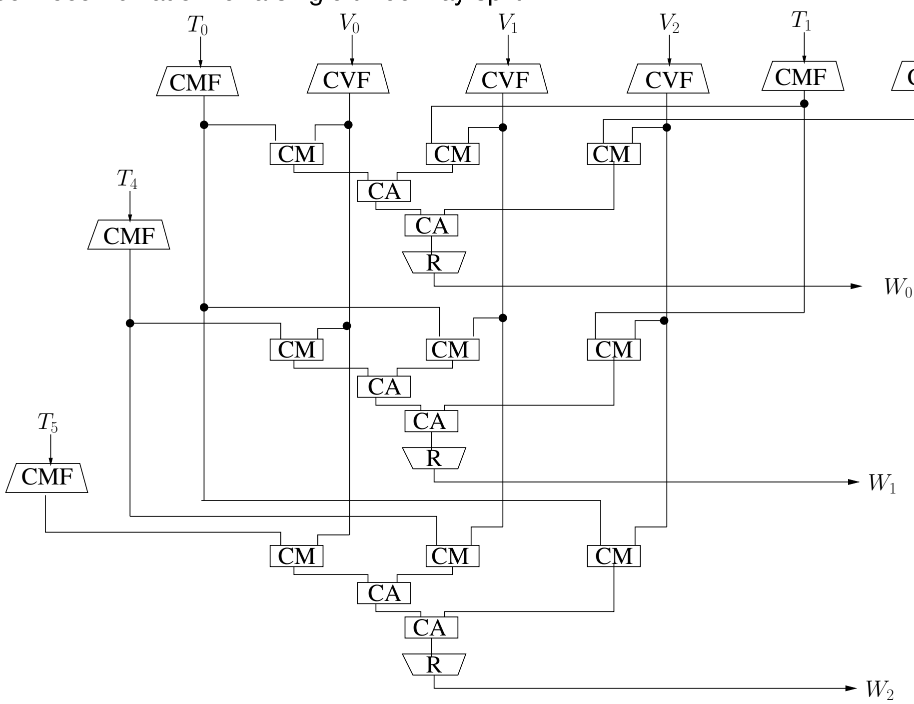
3.3 Recombination for a single three-way split TMVP multiplier

We now apply the method used in the previous section to the three-way split approach of Fan-Hasan multiplier. Specifically we want to compute a Toeplitz matrix vector product $W = T \cdot V$ where n, the size of T and V, is a power of 3. As in the two-way split case, we perform the first step of the three-way split approach using a direct computation of the matrix vector product.

$$T \cdot V = \begin{bmatrix} T_0 \cdot V_0 + T_1 \cdot V_1 + T_2 \cdot V_2 \\ T_3 \cdot V_0 + T_0 \cdot V_1 + T_1 \cdot V_2 \\ T_4 \cdot V_0 + T_3 \cdot V_1 + T_0 \cdot V_2 \end{bmatrix}. \quad (6)$$

We remark here that there are 5 matrices involved T_0, T_1, T_2, T_3, T_4 and 3 vectors V_0, V_1, V_2. We can design a multiplier architecture using a similar block recombination as it was done for two-way split multiplier. We perform the recursive component formation on the 5 matrices and the 3 vectors. We then multiply them using component multiplication. We put the addition block before the reduction block. The resulting architecture is depicted in Fig. 10.

Fig. 10. Block recombination for a single three-way split TMVP



We evaluate the complexity by counting the contribution of each elementary blocks in the number of XOR and AND gates. Note that in our complexity calculation, we have assumed that the library only contains two-input AND gates and two-input XOR gates for our implementation. The space complexity could be further reduced by making use of multiple input gates. We obtain the following expression for the number of XOR gates and the number of AND gates

$$\begin{aligned}\mathcal{S}_{\oplus} &= 5\mathcal{S}^{T}_{3,\oplus}(\tfrac{n}{3}) + 3\mathcal{S}^{V}_{3,\oplus}(\tfrac{n}{3}) + 3\mathcal{S}^{R}_{3,\oplus}(\tfrac{n}{3}) + 6\mathcal{S}^{A}_{3,\oplus}(\tfrac{n}{3}),\\ \mathcal{S}_{\otimes} &= 9\mathcal{S}^{M}_{3,\oplus}(\tfrac{n}{3}).\end{aligned}$$

Then using the formula in Table 3 and Remark 1, we obtain

$$\mathcal{S}_{\oplus}(n) = 4n^{\log_2(3)} - \frac{19n}{3} + 1 \quad \text{and} \quad \mathcal{S}_{\otimes}(n) = \frac{3}{2}n^{\log_2(3)}$$

and the time complexity is equal to $D_A + (3\log_3(n) + 1)D_X$.

3.4 Complexity comparison

In this section we compare the multipliers of Fig. 9 and Fig. 10 to other similar architectures in the literature. The complexity results are listed in Table 5.

As can be seen from the table, the Fan-Hasan architectures outperform CRT, Karatsuba and Winograd methods regarding both space and time complexities. In two-way split approach our proposed method

TABLE 5

Complexity comparison of subquadratic complexity multipliers

Method	Split	# AND	# XOR	Delay
CRT [1]	2	$5.17n^{1.6}+4n^{1.4}+O(n^{1.2})$	$5.17n^{1.6}+7n^{1.4}+O(n^{1.2})$	$O(n)D_X+4n^{0.4}D_A$
Karatsuba [11]		$n^{\log_2(3)}$	$6n^{\log_2(3)}-6n+k-1$	$D_A+(3\log_2(n)+1)D_X$
Fan-Hasan [3]		$n^{\log_2(3)}$	$5.5n^{\log_2(3)}-6n+0.5$	$D_A+2\log_2(n)D_X$
proposed		$1.33n^{\log_2(3)}$	$5.17n^{\log_2(3)}-7.5n+1.5$	$D_A+2\log_2(n)D_X$
CRT [1]	3	$5.17n^{1.6}+4n^{1.4}+O(n^{1.2})$	$5.17n^{1.6}+7n^{1.4}+O(n^{1.2})$	$O(n)D_X+4n^{0.4}D_A$
Winograd [15]		$n^{\log_3(6)}$	$5.33n^{\log_3(6)}-\frac{16n}{3}+k-1$	$D_A+(4\log_3(n)+1)D_X$
Fan-Hasan [3]		$n^{\log_3(6)}$	$4.8n^{\log_3(6)}-5n+\frac{1}{5}$	$D_A+3\log_3(n)D_X$
proposed		$1.5n^{\log_2(3)}$	$4n^{\log_2(3)}-6.33n+1$	$D_A+3\log_3(n)D_X$

presents the same time complexity as that of Fan-Hasan. It also matches the total number of gates used by Fan-Hasan. Since our proposal uses fewer number of XOR gates, it is more suitable for ASIC implementations as the area of an XOR gate is larger than that of an AND gate[1] in CMOS libraries.

Regarding the three-way split approach, it can be seen from the table that our proposal and the Fan-Hasan scheme present the least time complexities. Regarding space complexities our proposed method has the smallest area usage compared to all other architectures listed in the table.

4 Application to GHASH

In this section we present an application of the block recombination of two-TMVPs-and-add architecture presented in Subsection 3.1 to the GHASH function. GHASH is used Galois counter mode (GCM) [9] to compute the Message Authentification Code (MAC). GHASH is defined as

$$GHASH(C) = C_1H + C_2H^2 + \cdots C_NH^N$$

where $C = (C_1, \ldots, C_N)$ is such that $C_i \in \mathbb{F}_{2^n}$ and $H \in \mathbb{F}_{2^n}$. The element H remains unchanged for a single block of messages. The most costly computation for $GHASH$ is multiplication in $\mathbb{F}_{2^n}$. To speed up, two multiplications can be performed in parallel. We recall here a first method for parallel GHASH implementation presented in [14]. We then present our proposal for parallel implementation of GCM with smaller area requirements.

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

4.1 Parallel GHASH through polynomial decomposition in even/odd parts [14]

The authors in [14] have noticed that $GHASH(C)$ can be expressed as

$$\begin{aligned} GHASH(C) &= \overbrace{\left(C_2H^2 + C_4H^4 + \cdots + C_{N-4}(H^2)^{N/2-2} + C_{N-2}(H^2)^{N/2-1}\right)}^{C_{even}(H^2)} \\ &\quad + Y \underbrace{\left(C_1 + C_3H^2 + \cdots + C_{N-3}(H^2)^{N/2-2} + C_{N-1}(H^2)^{N/2-1}\right)}_{C_{odd}(H^2)}, \end{aligned}$$

where we assume that N is even. In other words if we call $C_{odd}(H^2)$ the odd part of $GHASH(C)$ and $C_{even}(H^2)$, the even part of $GHASH(C)$, then we have

$$C(H) = C_{even}(H^2) + HC_{odd}(H^2).$$

$C_{even}(H^2)$ and $C_{odd}(H^2)$ are independent GHASH computations using the constant H^2. They can be computed in parallel. Algorithm 2 computes $GHASH(C)$ in H using this approach.

Algorithm 2 ParallelEval

Require: $C = (C_1, \ldots, C_N)$ where $C_i \in \mathbb{F}_{2^n}$ and $H \in \mathbb{F}_{2^n}$

Ensure: $h = GHASH(C)$

$G \leftarrow H^2$

$h_{odd} \leftarrow 0$

$h_{even} \leftarrow 0$

for $i = N/2 - 1$ **to** 0 **do**

  $h_{even} \leftarrow h_{even} \times G + C_{2i+2}$

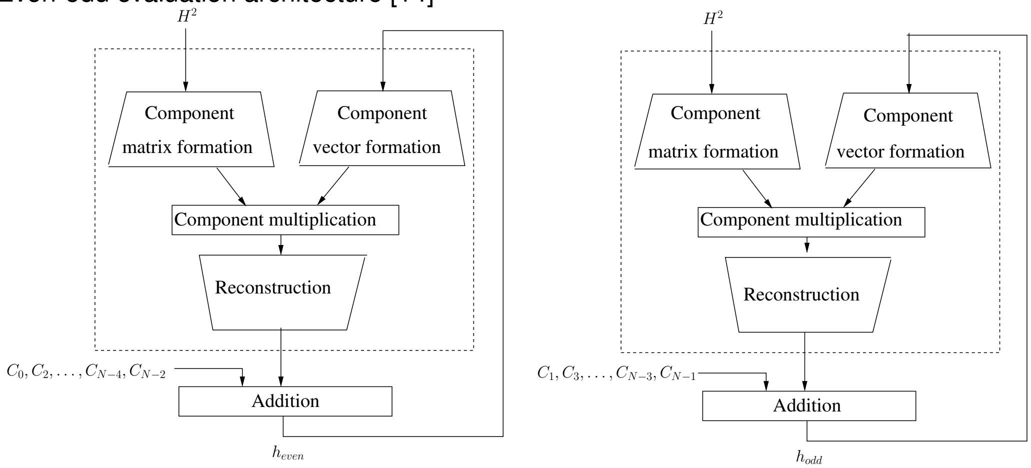
  $h_{odd} \leftarrow h_{odd} \times G + C_{2i+1}$

end for

return $h \leftarrow (Gh_{even} + Hh_{odd})$

A hardware architecture corresponding to Algorithm 2 is depicted in Fig. 11. This architecture requires two $\mathbb{F}_{2^n}$ multipliers in parallel which independently compute $C_{odd}(H^2)$ and $C_{even}(H^2)$. We have decomposed the two multipliers into elementary blocks in order to easily evaluate the space complexity.

Fig. 11. Even-odd evaluation architecture [14]



4.2 Two multiply and add approach

Our proposal to compute $GHASH(C)$ consists of using a *Most Significant Digit expression* of $GHASH(C)$

$$GHASH(C) = (((\cdots((C_N H + C_{N-1})H^2 + C_{N-2}H + C_{N-3})H^2 + \cdots)H^2 + C_4 H + C_3)H^2 + C_2 H + C_1)H.$$

This expression can be computed through a sequence of length $N/2$ of *two-multiply-and-add* in $\mathbb{F}_{2^n}$. Algorithm 3 computes $GHASH(C)$ using this method.

Algorithm 3 TwoMultAddEval

Require: $C = (C_1, \ldots, C_N)$ where $C_i \in \mathbb{F}_{2^n}$ and $H \in \mathbb{F}_{2^n}$

Ensure: $h = GHASH(C)$

  $G \leftarrow H^2$

  $h \leftarrow 0$

  for $i = N/2 - 1$ **to** 0 **do**

    $h \leftarrow h \times G + C_{2i+2}H + C_{2i+1}$

  end for

  $h \leftarrow h \times H$

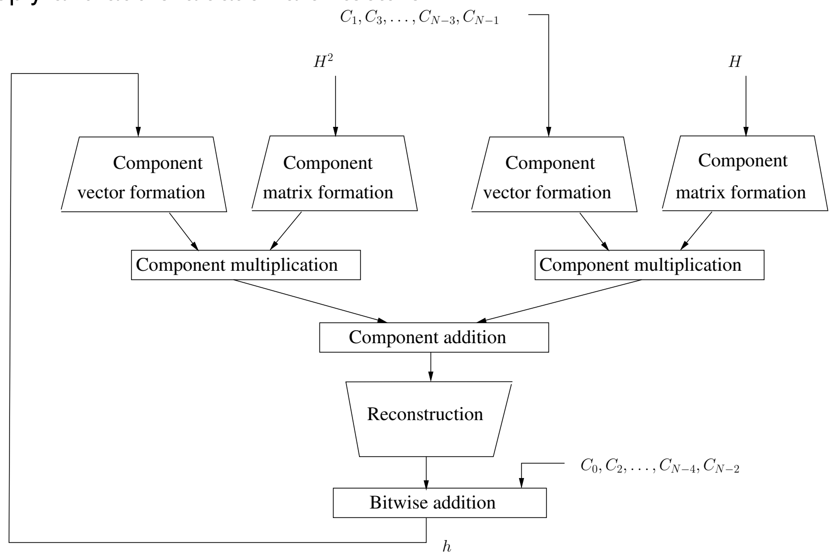
  return h

The corresponding hardware architecture uses the approach previously presented in Subsection 3.1. Specifically the two multipliers are recombined in order to reduce the space complexity.

4.3 Comparison of the two approaches

We now evaluate the complexity of the architecture shown in Fig. 11 and Fig. 12 using the block complexity of Table 3.

Fig. 12. Two-multiply-and-add evaluation architecture



- *Complexity of the architecture of Fig. 11.* The space complexity of the architecture of Fig. 11 is given by

$$\mathcal{S}_\oplus = 2\mathcal{S}_\oplus^T + 2\mathcal{S}_\oplus^V + 2\mathcal{S}_\oplus^R + 2n \text{ and } S_\otimes = 2\mathcal{S}_\otimes^M$$

All the multiplications are done with operand H^2. Consequently the component matrix formation of H^2 can be precomputed, and two CMF blocks can be removed from Fig. 11. The number of XORs becomes $\mathcal{S}_\oplus = 2\mathcal{S}^V + 2\mathcal{S}^R + 2\mathcal{S}^n + 2n$ and the number of AND remains the same. The delay is equal to the delay of a Fan-Hasan multiplier plus one D_X. The resulting complexities of the optimized version of Fig. 11 for the two-way split and three-way split are given in Table 6.

- *Complexity of architecture of Fig. 12.* The space complexity of the architecture of Fig. 12 is given by

$$\mathcal{S}_\oplus = 2\mathcal{S}_\oplus^T + 2\mathcal{S}_\oplus^V + \mathcal{S}_\oplus^R + \mathcal{S}_\oplus^A \text{ XOR gates and } S_\otimes = 2\mathcal{S}_\otimes^M \text{ AND gates.}$$

Again, the multiplications are done with operand H^2 and operand H at each time. Thus, the component matrix formation of H^2 and H can be precomputed, and the CMF blocks can be removed from Fig. 12. The resulting complexities of this optimized version of Fig. 12 are given in Table 6.

Table 6 shows that our architecture has the same delay as the architecture presented in [14] in each case while the area decreases by $n^{\log_2(3)} - n$ (resp. $n^{\log_3(6)} - n$) for the two-way split (resp. the three-way split) approach.

Remark 3. The authors in [14] presented their approach for k parallel multiplications. For the sake of simplicity we have focused on the case $k = 2$. Algorithm 3 and its corresponding architecture (Fig. 12) can be extended to k parallel multiplication and additions. For the general case, we would have similar results regarding the improvement on space complexity.

TABLE 6

Complexity comparison of GHASH architectures

Archi	Split	Space complexity		Time complexity
		#AND	#XOR	
[14]	2	$6n^{\log_2(3)} - 4n$	$2n^{\log_2(3)}$	$(2\log_2(n)+1)D_X + D_A$
Fig. 12	2	$5n^{\log_2(3)} - 3n$	$2n^{\log_2(3)}$	$(2\log_2(n)+1)D_X + D_A$
[14]	3	$6n^{\log_3(6)} - 4n$	$2n^{\log_3(6)}$	$(3\log_3(n)+1)D_X + D_A$
Fig. 12	3	$5n^{\log_3(6)} - 3n$	$2n^{\log_3(6)}$	$(3\log_3(n)+1)D_X + D_A$

5 Conclusion

The Toeplitz matrix vector multiplier of Fan and Hasan [3] can be decomposed into different independent blocks. In this paper we have used this decomposition to design an architecture which performs two-TMVPs-and-add with smaller area requirements. Moreover we have used this block recombination method for a single TMVP architecture. We have modified the first recursive computation in the Fan-Hasan multiplier and then recombine the resulting blocks. We again obtained a multiplier with a lower space complexity. Finally, we have applied our block recombination approach to develop an efficient parallel implementation of GHASH function of GCM.

References

[1] Jean-Claude Bajard, Laurent Imbert, and Graham A. Jullien. Parallel Montgomery Multiplication in GF(2^k) Using Trinomial Residue Arithmetic. In *IEEE Symposium on Computer Arithmetic*, pages 164–171, 2005.

[2] Aaron E. Cohen and Keshab K. Parhi. Implementation of scalable elliptic curve cryptosystem crypto-accelerators for GF(2^m). In *Proc. 13th Asilomar Conf. on Signals, Systems and Computers*, pages 471–477, 2004.

[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, September 2007.

[4] M.A. Hasan and V.K. Bhargava. Division and Bit-Serial Multiplication over GF(q^m). *IEE Proc. Computers and Digital Techniques*, pages 230–236, may 1992.

[5] T. Itoh and S. Tsujii. Structure of parallel multipliers for a class of fields GF(2^m). *Inform. Comp.*, 83:21–40, 1989.

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

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

[8] E. Mastrovito. VLSI Designs for Multiplication over Finite Fields $F(2^m)$. In *6th International Conference on Applied Algebra, Algebraic Algorithm and Error-Correcting Codes (AAECC-6)*, pages 297–309, 1988.

[9] David A. McGrew and John Viega. The security and performance of the galois/counter mode (gcm) of operation. In *INDOCRYPT*, pages 343–355, 2004.

[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] A. Reyhani-Masoleh and M.A. Hasan. A New Construction of Massey-Omura Parallel Multiplier over GF(2^m). *IEEE Trans. Computers*, 51(5):511–520, 2002.

[13] F. Rodriguez-Henriquez and C.K. Koç. Parallel multipliers based on special irreducible pentanomials. *IEEE Transaction on Computers*, 52(12):1535–1542, December 2003.

[14] Akashi Satoh. High-speed parallel hardware architecture for galois counter mode. In *ISCAS*, pages 1863–1866, 2007.

[15] B. Sunar. A generalized method for constructing subquadratic complexity GF(2^k) multipliers. *IEEE Transactions on Computers*, 53:1097–1105, 2004.

[16] B. Sunar and C. Koc. Mastrovito Multiplier for All Trinomials. *IEEE Trans. Computers*, 48(5):522–527, 1999.

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

Appendix

When $n = 1$, we have $U = R(\hat{U}) = \hat{U}$ for all U which implies that $R(\hat{W}) + R(\hat{W}') = \hat{W} + \hat{W}' = R(\hat{W} + \hat{W}')$. Let us now show it for $n = 3^s$ assuming that (4) is true for $n = 3^{s-1}$. We decompose $\hat{W}$ and $\hat{W}'$ in six parts of size $n^{\log_3(6)}/6$

$$\begin{aligned}\hat{W} &= [\hat{W}_0, \hat{W}_1, \hat{W}_2, \hat{W}_3, \hat{W}_4, \hat{W}_5],\\ \hat{W}' &= [\hat{W}'_0, \hat{W}'_1, \hat{W}'_2, \hat{W}'_3, \hat{W}'_4, \hat{W}'_5].\end{aligned}$$

We then apply the definition of R given in Table 1 to these expressions of $\hat{W}$ and $\hat{W}'$

$$\begin{aligned}R(\hat{W}) &= [R(\hat{W}_0) + R(\hat{W}_3) + R(\hat{W}_4), R(\hat{W}_1) + R(\hat{W}_3) + R(\hat{W}_5), R(\hat{W}_2) + R(\hat{W}_4) + R(\hat{W}_5)],\\ R(\hat{W}') &= [R(\hat{W}'_0) + R(\hat{W}'_3) + R(\hat{W}'_4), R(\hat{W}'_1) + R(\hat{W}'_3) + R(\hat{W}'_5), R(\hat{W}'_2) + R(\hat{W}'_4) + R(\hat{W}'_5)]\end{aligned}$$

Using these expressions we get for $R(\hat{W}) + R(\hat{W}')$

$$\begin{aligned}R(\hat{W}) + R(\hat{W}') = {} & [R(\hat{W}_0) + R(\hat{W}'_0) + R(\hat{W}_3) + R(\hat{W}'_3) + R(\hat{W}_4) + R(\hat{W}'_4),\\ & R(\hat{W}_1) + R(\hat{W}'_1) + R(\hat{W}_3) + R(\hat{W}'_3) + R(\hat{W}_5) + R(\hat{W}'_5),\\ & R(\hat{W}_2) + R(\hat{W}'_2) + R(\hat{W}_4) + R(\hat{W}'_4) + R(\hat{W}_5) + R(\hat{W}'_5)].\end{aligned}$$

We now apply the inductive hypothesis which gives that $R(\hat{W}_i) + R(\hat{W}'_i) = R(\hat{W}_i + \hat{W}'_i)$ for each $0 \leq i \leq 5$ and we get

$$\begin{aligned}R(\hat{W}) + R(\hat{W}') = {} & [R(\hat{W}_0 + \hat{W}'_0) + R(\hat{W}_3 + \hat{W}'_3) + R(\hat{W}_4 + \hat{W}'_4),\\ & R(\hat{W}_1 + \hat{W}'_1) + R(\hat{W}_3 + \hat{W}'_3) + R(\hat{W}_5 + \hat{W}'_5),\\ & R(\hat{W}_2 + \hat{W}'_2) + R(\hat{W}_4 + \hat{W}'_4) + R(\hat{W}_5 + \hat{W}'_5)].\end{aligned}$$

This last expression is in fact the recursive formula of R applied to

$$\hat{W} + \hat{W}' = [\hat{W}_0 + \hat{W}'_0, \hat{W}_1 + \hat{W}'_1, \hat{W}_2 + \hat{W}'_2, \hat{W}_3 + \hat{W}'_3, \hat{W}_4 + \hat{W}'_4, \hat{W}_5 + \hat{W}'_5].$$

This ends the proof for the three-way split case.