Journal of Computers Vol. 31 No. 3, 2020, pp. 11-26
doi:10.3966/199115992020063103002

Subquadratic Complexity Gaussian Normal Basis Multiplier with Subquadratic and Quadratic Computation Approach

Che Wun Chiou[1], Chiou-Yng Lee[2], Yuh-Sien Sun[3*], Cheng-Min Lee[3],
Shih Shng Chen[4], Jim-Min Lin[5], Tai-Pao Chuang[1]

[1] Department of Computer Science and Information Engineering, Chien Hsin University of Science and Technology, Taoyuan City 32097, Taiwan
{cwchiou, tpchuang}@uch.edu.tw

[2] Department of Computer Information and Network Engineering, Lunghwa University of Science and Technology, Taoyuan City 33306, Taiwan
PP010@mail.lhu.edu.tw

[3] Department of Electronic Engineering, Chien Hsin University of Science and Technology, Taoyuan City 32097, Taiwan
{sunys, cmlee}@uch.edu.tw

[4] Department of Cultural Creativity and Design, Nan Kai University of Technology, Nantou County 54243, Taiwan
sostc@nkut.edu.tw

[5] Department of Information Engineering and Computer Science, Feng Chia University, Taichung City 407, Taiwan
jimmy@fcu.edu.tw

Received 30 August 2018; Revised 1 July 2019; Accepted 15 July 2019

Abstract. Finite field multiplication over GF(2^m) is the most important arithmetic operation in elliptic curve cryptography. Efficient hardware and software implementations of finite field multiplication are important and necessary. In the past, the Toeplitz matrix-vector product (TMVP) approach was used widely for subquadratic space complexity finite field multipliers. However, the TMVP approach is not effective for core multipliers of such subquadratic space complexity finite field multipliers. Therefore, this study will present a novel subquadratic space complexity type-t Gaussian normal basis (GNB) multiplier, which uses a non-TMVP core multiplier instead of the TMVP core multiplier found in existing approaches. The space complexity of the proposed type-t GNB multiplier is 26% lower than that in the best existing subquadratic space complexity GNB multipliers and the time complexity is 17% lower.

Keywords: elliptic curve cryptography, finite field, Gaussian normal basis, subquadratic computation complexity multiplier, Toeplitz matrix-vector product

1 Introduction

Rapid expansion in the use of smart phones has provided greater opportunity for participation in mobile commerce (M-commerce). Elliptic curve cryptography (ECC) [1-2] is an emerging system used to ensure the security of information related to M-commerce, particularly when transactions are conducted using resource-constrained smart phones. Arithmetic operations over GF(2^m) are widely used in ECC and pairing-based cryptography [3]. ECC uses a key far smaller than that required for RSA cryptosystems [4].

[*] Corresponding Author

For example, an ECC with a 160-bit key provides roughly the same security as RSA with a 1024-bit key. ECC applications employ scalar multiplications, which are realized by several point additions and point doublings along elliptic curves. These scalar multiplications use either projective coordinates or affine coordinates over an extension binary field GF(2^m) or prime field GF(p). Point addition within affine coordinates over GF(2^m) involves field operations, such as addition, squaring, multiplication, and inversion. Projective coordinates are suitable for high-performance ECC designs, because each point addition involves field operations, such as addition, squaring, and multiplication, while the inversion operation is performed only for coordinate transformations from projective coordinates to affine coordinates. For example, NIST and ANSI include recommended finite fields for use with ECDSA [5-6]. In binary fields, addition and squaring are relatively simple operations, whereas multiplication incurs considerable complexity. This has necessitated the development of efficient multipliers for use over large finite fields when using resource-constrained devices.

Multiplication in GF(2^m) depends heavily on the representation of the field element. There are three common bases used for the representation of field elements: polynomial basis (PB) [7-15], dual basis (DB) [16-21], and normal basis (NB) [22-35]. NB provides nearly cost-free squaring operations, which are easily carried out by cyclically shifting its binary representation. This makes NB multipliers highly efficient in performing square operations of multiplicative inversion, squaring, and exponentiation. Unfortunately, performing NB multiplication can be difficult because it needs very high XOR gate complexity. A number of special classes of NB can be selected to simplify NB multiplication. The special class referred to as optimal NB (ONB) [27] provides the lowest space complexity. Unfortunately, only two types of ONB, type-1 and type-2, have been reported in the literature. Gaussian NB (GNB) is another special class of NB, which features low hardware complexity. All positive integers except those divisible by eight have GNB [36]. Type-1 and type-2 ONBs are same as type-1 and type-2 GNBs. GNB is now included in several standards, such as IEEE Standard 1363-2000 [5], FIPS 186-2 [37], ISO 11770-3 [38], and ANSI X9.62 [6].

Many architectures have been developed for the multipliers in GF(2^m*)*, including bit-parallel, bit-serial, digit-serial, and hybrid. Bit-parallel multipliers concurrently generate all result bits, which are synchronized within a single clock cycle and therefore incur a shorter execution time but require higher hardware costs. The hardware costs of bit-serial multipliers is somewhat reduced; however, they require longer execution time. Digit-serial multipliers give d ($1 \le d \le m$) result bits within a single clock cycle with a trade-off between time and space complexities. Subquadratic space complexity designs have been incorporated in hybrid multipliers based on the divide-and-conquer approach. Many methods based on the divide-and-conquer method have been developed, including the Karatsuba algorithm [39], the Toom-Cook algorithm [15], Toeplitz matrix-vector (TMVP) decomposition [12], and Fourier transform [40].

The first NB multiplier with parallel-in serial-out structure was developed by Massey and Omura [22] in 1986. Many variations of Massey-Omura NB multipliers have since been developed [23, 25-26]. Reyhani-Masoleh [24] used the symmetry property of Gaussian period to develop a non-systolic type-t GNB multiplier capable of outperforming conventional basis multipliers. Chiou et al. [34] provided a low-complexity systolic array for bit-parallel Gaussian NB multipliers using redundant polynomial ring. Lee and Chiou [29] employed the Hankel matrix-vector product approach in the development of a scalable GNB multiplier. Azarderakhsh and Reyhani-Masoleh [32] derived a hybrid-double multiplier to speed up the exponentiation and point multiplication in public-key cryptosystems. Recently, a new approach, termed Toeplitz matrix-vector product (TMVP), is employed by many researchers [11-12, 21, 28, 33, 35, 41-42] for developing subquadratic space complexity multipliers. Studies in [11-12, 41-42] are referred to PB. Researches in [35] and [28] considered in the NB. Authors in [21] are focused on the double basis. Fan and Hasan [12] used the Toeplitz matrix-vector product (TMVP) approach to design subquadratic space complexity PB, shifted PB, DB, and triangular basis multipliers. Lee et al. [11] used the (b,2)-way Karatsuba decomposition algorithm in the design of subquadratic space complexity scalable shifted PB and generalized PB multipliers. Xie et al. [41] used two-way TMVP method and optimization techniques to design a subquadratic complexity systolic PB multiplier for better area-time complexity than existing related works. Pan et al. [42] firstly proposed a digit-serial systolic PB multiplier covering all irreducible polynomials using TMVP approach. Leone [35] proposed a subquadratic space complexity type-1 ONB multiplier via recursive application of the Karatsuba algorithm. Fan and Hasan [28] applied the TMVP approach to the design of subquadratic computational complexity type-1 and type-2 ONB multipliers. Yang et al. [33] employed the Tensor product and TMVP approaches in the development of a subquadratic space complexity digit-serial GNB multiplier. Pan et al. [21] used the TMVP approach in deriving a low-latency digit-serial systolic double basis multiplier. However, the core circuits of multipliers using TMVP method are only logical AND operations not the complicated mathematical multiplications. If the mathematical multiplications are empolyed for computing core circuits, thus it consumes much time. In this paper, we propose a subquadratic space complexity bit-parallel type-*t* GNB. We employed the TMVP approach to decompose an *m*×*m* matrix and *m*-bit vector into 2×2 matrix and 2-bit vector. An AND-XOR circuit is used in the TMVP multiplier instead of an XOR-AND-XOR [12, 21, 28, 33] for the multiplication functions of the 2×2 matrix and a 2-bit vector.

The major contributions of this study are listed as follows:

(1) The first TMVP multiplier proposed in this study uses non-TMVP core multipliers rather than the TMVP core multipliers found in most existing subquadratic complexity multipliers. This leads to 26% reduction in space complexity and a 17% reduction in time complexity.

(2) This study presents an effective method with which to simplify general type-*t* GNB multiplication.

(3) Details of the hardware circuits used in the proposed multiplier are also provided.

The remainder of this paper is organized as follows. In Section 2, we present a brief preliminary discussion of GNB and the Toeplitz matrix-vector product. In Section 3, we outline the proposed subquadratic GNB multiplier using a non-TMVP core multiplier. Section 4 outlines a simplified subquadratic GNB multiplier. Section 5 presents a comparison of the proposed GNB multiplier with similar existing multipliers. Conclusions are drawn in Section 6.

2 Background

Normal basis, Gaussian normal basis, and Toeplitz matrix-vector product are briefly reviewed in the following.

2.1 Normal Basis and Gaussian Normal Basis

For a finite field GF(2^m) over GF(2) for any positive integer m, there will always exist a normal basis $\Lambda = \left\{ \alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, ..., \alpha^{2^m-1} \right\}$, where α is a normal element. Any two elements *A* and *B* $\in$ GF(2^m) can be represented as

$$A = (a_0, a_1, a_2, ..., a_{m-1}) = \sum_{i=0}^{m-1} a_i \alpha^{2^i}, \text{ and } B = (b_0, b_1, b_2, ..., b_{m-1}) = \sum_{i=0}^{m-1} b_i \alpha^{2^i},$$

where a_i and b_i $\in$GF(2) for $0 \le i \le m-1$. The major feature of the normal basis is as follows:

Property-1:

$$\begin{aligned} A^{2^r} &= (a_{m-r}, a_{m-r+1}, ..., a_{m-1}, a_0, a_1, ..., a_{m-r-1}) \\ &= a_{m-r}\alpha^{2^0} + a_{m-r+1}\alpha^{2^1} + ... + a_{m-1}\alpha^{2^{r-1}} + a_0\alpha^{2^r} + a_1\alpha^{2^{r+1}} + ... + a_{m-r-1}\alpha^{2^{m-1}}, \end{aligned} \quad \text{for } 1 \le r \le m.$$

Two important features for any elements in GF(2^m) are listed as follows:

Property-2: $A^{2^m} = A$,

Property-3: $(A+B)^2 = A^2 + B^2$.

Property-1 shows that the squaring of element *A* in normal basis is simply a right cyclic shift in its coordinates, and therefore incurs no hardware cost. Type-*t* GNB is defined as follows:

Definition 1: Normal basis $\Lambda = \left\{ \alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, ..., \alpha^{2^m-1} \right\}$ is referred to as type-*t* Gaussian normal basis if *p*=*mt*+1 is a prime number and gcd(*mt/k,m*)=1, where *k* is the multiplication order of 2 modulo *p*.

It should be noted that GNBs exist for any positive integer *m*, except when *m* is divisible by 8. In this paper, only odd values of *m* are considered. No generality is lost by assuming odd values for *m* because the five *m* values recommended by NIST for ECDSA ($\{163, 233, 283, 409, 571\}$) are all odd values.

Type-*t* GNB (*t* is an integer and *t*≧1) has the following attributes:

Property-4: $\alpha = \sum_{i=0}^{t-1} \gamma^{2^{mi}}$,

Property-5: $\gamma^{mt+1} = \gamma^{(mt+1) \bmod (mt+1)} = 1$,

where γ is primitive $(mt+1)^{\text{th}}$ root of unity in GF(2^m). Then, α is referred to as a Gaussian period of type (m,t).

2.2 Toeplitz Matrix-vector Product

In this subsection, we briefly review the Toeplitz matrix-vector product algorithm [12, 21, 28], which is applied for the computation of DB, SPB, and normal bases.

Definition 2: An $n \times n$ matrix **H** can be referred to as a Toeplitz matrix if it satisfies the following relationship: $h(i,j)=h(i+1,j+1)$, where $0 \le i, j \le n-2$ and $h(i,j)$ represents the elements in row i and column j within matrix **H**.

Definition 3: An $n \times n$ Toeplitz matrix **H** with elements $h(i,j)$ in row i and column j can be defined by a sequence of $2n$-1 entities, such as $\underline{h} = (h(n\text{-}1,0), h(n\text{-}2,0),\ldots, h(0,0), h(0,1),\ldots, h(0,n\text{-}1))$.

A Toeplitz matrix has the following two properties:

Property-6: Any single $r \times r$ submatrix of an $n \times n$ Toeplitz matrix **H** also qualifies as a Toeplitz matrix for $1 \le r \le n$.

Property-7: An $n \times n$ Toeplitz matrix **H** can be determined by the $2n$-1 comprising elements in the first column and the first row. Thus, adding two $n \times n$ Toeplitz matrices requires only $3n/2-1$ additions (i.e., $3n/2-1$ XOR operations) [12].

Due to the special structure of the Toeplitz matrix, a number of elements involved in the addition of two Toeplitz matrices can be reused. Thus, only $3n/2-1$ elements are necessarily generated. Let **V** be an n×1 column vector. The product **W**=**H**×**V** is referred to as the Toeplitz matrix-vector product (TMVP). According to [12, 21], the computation of subquadratic complexity in a TMVP using the two-way split approach can be described as follows:

Let n be an even number. The $n \times n$ Toeplitz matrix **H** and the $n \times 1$ vector **V** can be decomposed into four submatrices and two subvectors. The product **W**=**HV** could be rewritten as

$$\mathbf{W} = \mathbf{H} \times \mathbf{V} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{H}_0 & \mathbf{H}_1 \end{bmatrix} \begin{bmatrix} \mathbf{V}_0 \\ \mathbf{V}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1(\mathbf{V}_0+\mathbf{V}_1)+(\mathbf{H}_1+\mathbf{H}_2)\mathbf{V}_1 \\ \mathbf{H}_1(\mathbf{V}_0+\mathbf{V}_1)+(\mathbf{H}_0+\mathbf{H}_1)\mathbf{V}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1+\mathbf{P}_2 \\ \mathbf{P}_1+\mathbf{P}_0 \end{bmatrix}, \tag{1}$$

where $\mathbf{H}_0$,$\mathbf{H}_1$, and $\mathbf{H}_2$ are $n/2 \times n/2$ Toeplitz matrices, $\mathbf{V}_0$ and $\mathbf{V}_1$ are $n/2 \times 1$ vectors, and $\mathbf{P}_0 = (\mathbf{H}_0+\mathbf{H}_1)\ \mathbf{V}_0$, $\mathbf{P}_1 = \mathbf{H}_1(\mathbf{V}_0+\mathbf{V}_1)$, and $\mathbf{P}_2 = (\mathbf{H}_1+\mathbf{H}_2)\mathbf{V}_1$.

According to Eq. (1), the number of multiplications can be reduced from n^2 to $3/4n^2$, while increasing the number of additions from $n(n\text{-}1)$ to $n(n+1)$. The following three steps are required to compute Eq. (1).

Step-1: Evaluation: This step involves two components: component matrix point (CMP) and component vector point (CVP). These two components are defined as

$$\text{CMP}(\mathbf{H})=(\mathbf{H}_0+\mathbf{H}_1, \mathbf{H}_1, \mathbf{H}_1+\mathbf{H}_2), \text{ and} \tag{2}$$

$$\text{CVP}(\mathbf{V})=(\mathbf{V}_0,\mathbf{V}_0+\mathbf{V}_1,\mathbf{V}_1). \tag{3}$$

Step-2: Point-wise multiplication (PWM): Based on the evaluation results in Step-1, PWM could be performed as

$$\begin{aligned} &\text{PWM}(\text{CMP}(\mathbf{H}), \text{CVP}(\mathbf{V})) \\ &=(\mathbf{P}_0(=(\mathbf{H}_0+\mathbf{H}_1)\mathbf{V}_0),\ \mathbf{P}_1(=\mathbf{H}_1(\mathbf{V}_0+\mathbf{V}_1)),\ \mathbf{P}_2(=(\mathbf{H}_1+\mathbf{H}_2)\mathbf{V}_1)). \end{aligned} \tag{4}$$

Step-3: Final Reconstruction (FR): Based on results in Step-2, FR could be carried out as

$$\text{FR}(\mathbf{P}) = [\mathbf{P}_1+\mathbf{P}_2, \mathbf{P}_1+\mathbf{P}_0]^{\text{Tr}}, \tag{5}$$

where Tr is the transpose operation.

We can use three-step operation recursively to implement the subquadratic TMVP multiplier, as shown in Fig. 1. The function unit CMP uses Eq. (2) to generate the set CMP(**H**)={**H**$_0$+**H**$_1$, **H**$_1$, **H**$_1$+**H**$_2$} which involves 3/2*n*-1 XOR gates. CVP uses Eq. (3) to transform the set {**V**$_0$,**V**$_0$+**V**$_1$, **V**$_1$}, which involves *n/*2 XOR gates. PWM uses Eq. (4) to perform the point-wise multiplication on sets CMP(**H**) and CVP(**V**) in order to obtain three sub-TMVPs (**P**$_0$, **P**$_1$, and **P**$_2$), which require $3\times\left(\frac{n}{2}-1\right)\times\frac{n}{2}$ XOR gates and $3\times\frac{n}{2}\times\frac{n}{2}$ AND gates. FR uses *n* XOR gates to reconstruct Eq. (5). Therefore, we can use these three operations recursively to derive the subquadratic TMVP multiplier. In [12, 21, 33], if $n=2^i\ \left(i\geq 1\right)$, then the two-way TMVP decomposition is estimated as $delays\left(TMVP\right)=T_A+\left(2\log_2 n\right)T_X$, where *delays(TMVP)*, T_A, and T_X represent the delay of the TMVP multiplier, the delay of the 2-input AND gate, and the delay of the 2-input XOR gate, respectively. The TMVP multiplier for multiplying an $n\times n$ Toeplitz matrix **H** and an $n\times 1$ vector **V** requires $3n^2/4+3n/2$ 2-input XOR gates and $3n^2/4$ 2-input AND gates.

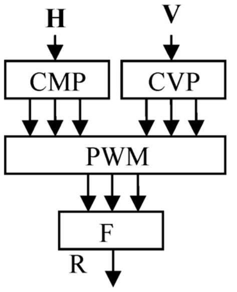


Fig. 1. The two-way split TMVP multiplier in [21]

2.3 Gaussian Normal Basis Multiplication

Let any two elements *A* and *B* of GF(2^m) be represented by type-*t* GNB as follows:

$$A=\sum\nolimits_{i=0}^{m-1}a_i\alpha^{2^i}\text{, and }B=\sum\nolimits_{i=0}^{m-1}b_i\alpha^{2^i},$$

where a_i and $b_i\ \in\mathrm{GF(2)}$ for $0\leq i\leq m-1$. Let *C* be their product; i.e., $C=A\times B$. Based on Property-4, element *A* can be represented as:

$$A=\sum\nolimits_{i=0}^{m-1}a_i\left(\sum\nolimits_{j=0}^{t-1}\gamma^{2^{mj}}\right)^{2^i}=\sum\nolimits_{i=0}^{m-1}a_i\left(\sum\nolimits_{j=0}^{t-1}\gamma^{2^{mj+i}}\right)=\sum\nolimits_{i=0}^{m-1}\sum\nolimits_{j=0}^{t-1}a_i\gamma^{2^{mj+i}}.$$

According to Property-5, we obtain

$$\gamma\times\gamma^i=\begin{cases}\gamma^{i+1} & \text{if } i\neq p-1\\ \gamma^0 & \text{if } i=p-1\end{cases}.$$

Thus, the normal basis $\Lambda=\left\{\ \alpha^{2^0},\alpha^{2^1},\alpha^{2^2},...,\alpha^{2^{m-1}}\right\}$ can be transformed into an extended polynomial basis $\Lambda^*=\left\{\ \gamma^0,\gamma^1,\gamma^2,...,\gamma^{p-1}\right\}$ and element *A* can be expressed as:

$$A=\sum\nolimits_{w=0}^{p-1}a_{F(w)}\gamma^w,$$

where $a_{F(0)}=0,\ F\left(w\right)=i$ and $w=2^{mj+i}\bmod p$ for $0\leq i\leq m-1$ and $0\leq j\leq t-1$.

Similarly, elements *B* and *C* are represented as follows:

$$B=\sum\nolimits_{w=0}^{p-1}b_{F(w)}\gamma^w\text{ and }C=\sum\nolimits_{w=0}^{p-1}c_{F(w)}\gamma^w\text{, where }b_{F(0)}=0.$$

Product *C* is calculated using the following equation:

$$C=\left(\sum_{w=0}^{p-1}a_{F(w)}\gamma^{w}\right)\times\left(\sum_{w=0}^{p-1}b_{F(w)}\gamma^{w}\right)=\sum_{w=0}^{p-1}c_{F(w)}\gamma^{w},$$

where $c_{F(w)}=\sum_{e=0}^{p-1}a_{F(\langle p-w\rangle)}b_{F(e)}$ for $0\le w\le p-1$.

Product *C* can also be computed using the matrix-vector form, as follows:

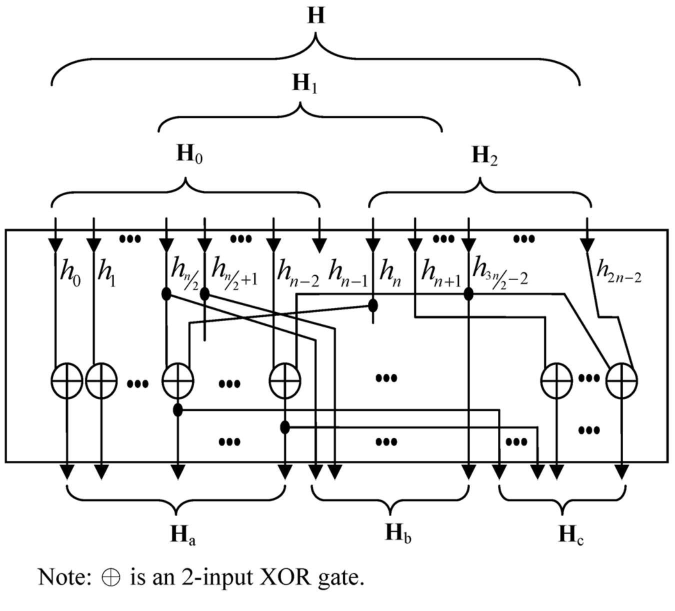
$$\begin{bmatrix} c_{F(0)} \\ c_{F(1)} \\ c_{F(2)} \\ \cdots \\ c_{F(p-1)} \end{bmatrix}=\begin{bmatrix} a_{F(0)} & a_{F(1)} & a_{F(2)} & \cdots & a_{F(p-1)} \\ a_{F(p-1)} & a_{F(0)} & a_{F(1)} & \cdots & a_{F(p-2)} \\ a_{F(p-2)} & a_{F(p-1)} & a_{F(0)} & \cdots & a_{F(p-3)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{F(1)} & a_{F(2)} & a_{F(3)} & \cdots & a_{F(0)} \end{bmatrix}\times\begin{bmatrix} b_{F(0)} \\ b_{F(1)} \\ b_{F(2)} \\ \cdots \\ b_{F(p-1)} \end{bmatrix}. \quad \textbf{(6)}$$

It should be noted that the above equation is a Toeplitz matrix-vector product. Thus, the two-way splitting approach of TMVP can be applied recursively to derive the multiplication with subquadratic computation complexity.

3 Proposed Subquadratic GNB Multiplier Using Non-TMVP Core Multiplier

Many researchers [12, 21, 28] have used recursive TMVP for the design of finite field multipliers in GF(2^m) with subquadratic space and/or time complexities. However, the TMVP core multiplier used for multiplying the 2×2 matrix and 1×2 vector is a time- and memory-consuming circuit. To overcome this problem, we develop a non-TMVP core multiplier having the AND-XOR structure for multiplying the 2×2 matrix and 1×2 vector for subquadratic GNB multipliers.

Based on Eq. (6), the GNB multiplication of $C=A\times B$ is a Toeplitz matrix-vector product, which means that it can be computed using a two-way split TMVP multiplier. The two-way split TMVP multiplier requires four hardware operators: CMP_n, CVP_n, MUL, and FR_n. The operator CMP_n inputs an $n\times n$ Toeplitz matrix and outputs three $\frac{n}{2}\times\frac{n}{2}$ Toeplitz matrices, in accordance with Eqs. (1) and (2). The circuit design of CMP_n is presented in Fig. 2. The operator CMP_n inputs an $n\times 1$ vector and outputs three $n/2\times 1$ vectors, in accordance with Eqs. (1) and (3). The design of the hardware circuit is presented in Fig. 3. The operator MUL multiplies a 2×2 Toeplitz matrix and a 2×1 vector to obtain a 2×1 vector, in accordance with Eq. (4). Fig. 4 illustrates the hardware implementation. MUL is the core multiplier proposed for the multiplication of a 2×2 Toeplitz matrix and a 2×1 vector.



Note: ⊕ is an 2-input XOR gate.

Fig. 2. Circuit design of CMP_n

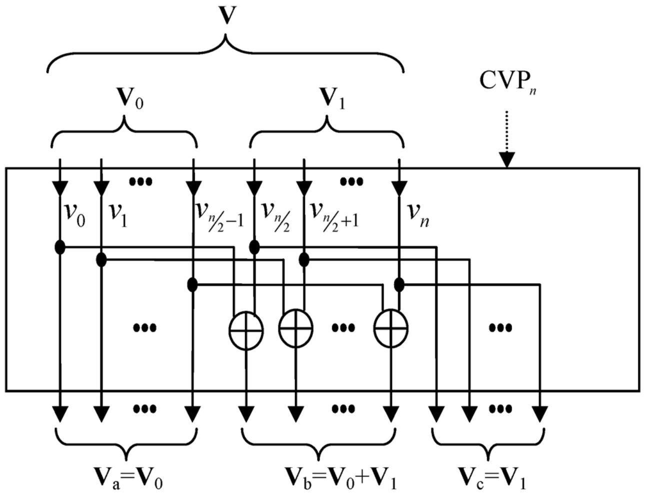


Fig. 3. Circuit design of CVP_n

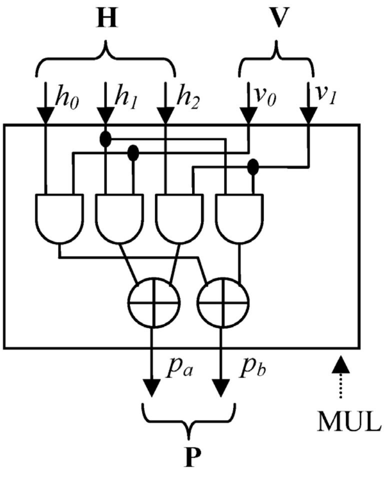


Fig. 4. Circuit design of MUL

It should be noted that the proposed core multiplier MUL does not follow the TMVP design found in [12, 21, 28, 33] (as shown in Fig. 5). The proposed core multiplier is an AND-XOR structure which only requires four 2-input AND gates and two 2-input XOR gates, whereas the traditional TMVP core multiplier requires three 2-input AND gates and five 2-input XOR gates. Using CMOS technology, a 2-input AND gate and a 2-input XOR gate require 6 and 8 transistors, respectively. Thus, the proposed core multiplier requires 40 transistors while the traditional TMVP core multiplier would require 58. Thus, the proposed core multiplier reduces transistor overhead by 31%, compared to the TMVP core multipliers in [12, 21, 28, 33]. Furthermore, the proposed core multiplier requires only one 2-input AND gate delay and one 2-input XOR gate delay, whereas the TMVP core multiplier requires one 2-input AND gate delay and two 2-input XOR gate delays. Using a chip fabricated based on the NanGate Library Creator and 45-nm FreePDK process design kit from North Carolina State University (NCSU) [43], the proposed core multiplier requires 0.07ns propagation delay whereas the traditional TMVP core multiplier requires 0.12ns in the case where the Input Transition=0.0012ns and Load Capacitance=0.3656 fF. Thus, the proposed core multiplier reduces propagation delay by approximately 41%, compared to the traditional TMVP core multiplier. Operator FR_{2_n} summarizes three input $n\times 1$ vectors and gives the resulting $2n\times 1$ vector in accordance with Eq. (5). The hardware circuit is outlined in Fig. 6. The proposed subquadratic GNB multiplier used for $C=A\times B$ is presented in Fig. 7. This proposed multiplier comprises $\lceil \log_2 n \rceil - 1$, 1, and $\lceil \log_2 n \rceil - 1$ layers of CMP, MUL, and FR, respectively, where $\lceil x \rceil$ is the ceiling function of x. In the proposed multiplier, the MUL replaces CMP$_2$, CVP$_2$, PWM$_2$, and FR$_2$ found in existing TMVP multipliers. For the sake of clarity, the traditional TMVP multiplier is presented in Fig. 8. The function block (dashed line) which involves CMP_2, CVP_2, PWM_2, and FR_2 in Fig. 8, is replaced by the block (dashed line) which consists of MULs in Fig. 7. In other words, the circuit in Fig. 4 of the proposed multiplier replaces the circuit in Fig. 5 of existing TMVP multipliers.

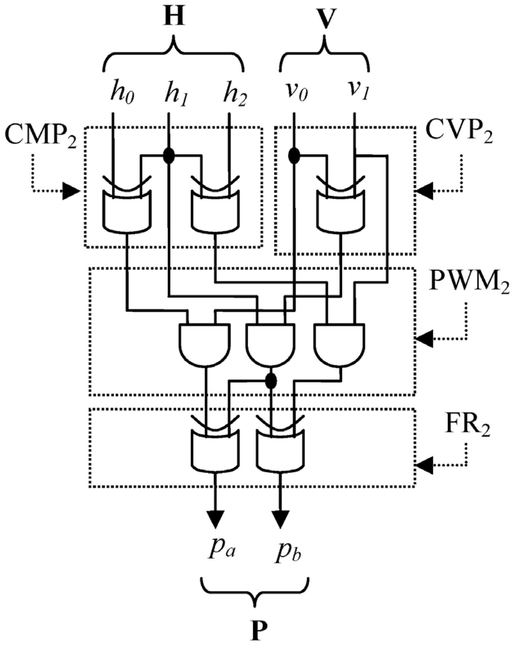


Fig. 5. Circuit design of TMVP MUL

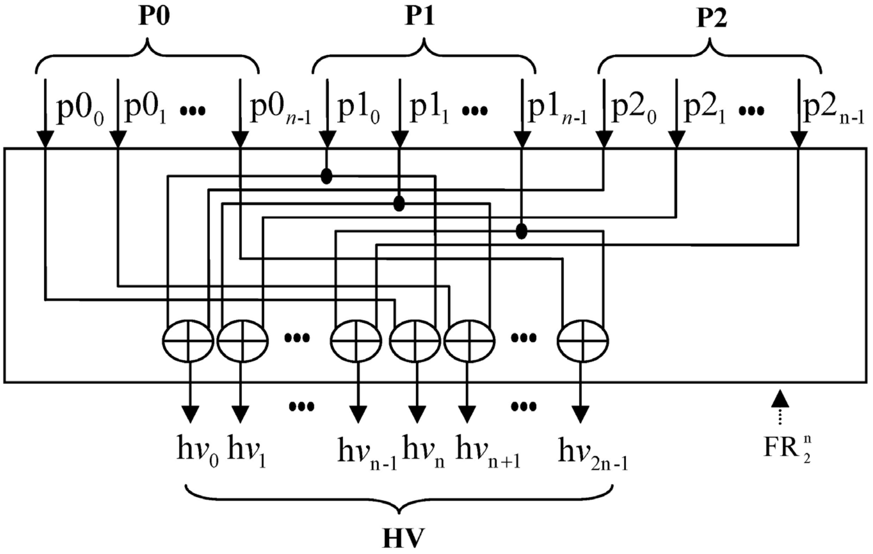

P0
P1
P2
$p0_0$ $p0_1$ ••• $p0_{n-1}$
$p1_0$ $p1_1$ ••• $p1_{n-1}$
$p2_0$ $p2_1$ ••• $p2_{n-1}$
•••
•••
hv_0 hv_1 hv_{n-1} hv_n hv_{n+1} hv_{2n-1}
FR_2^n
HV

Fig. 6. Circuit design of $\mathbf{FR}_{2n}$

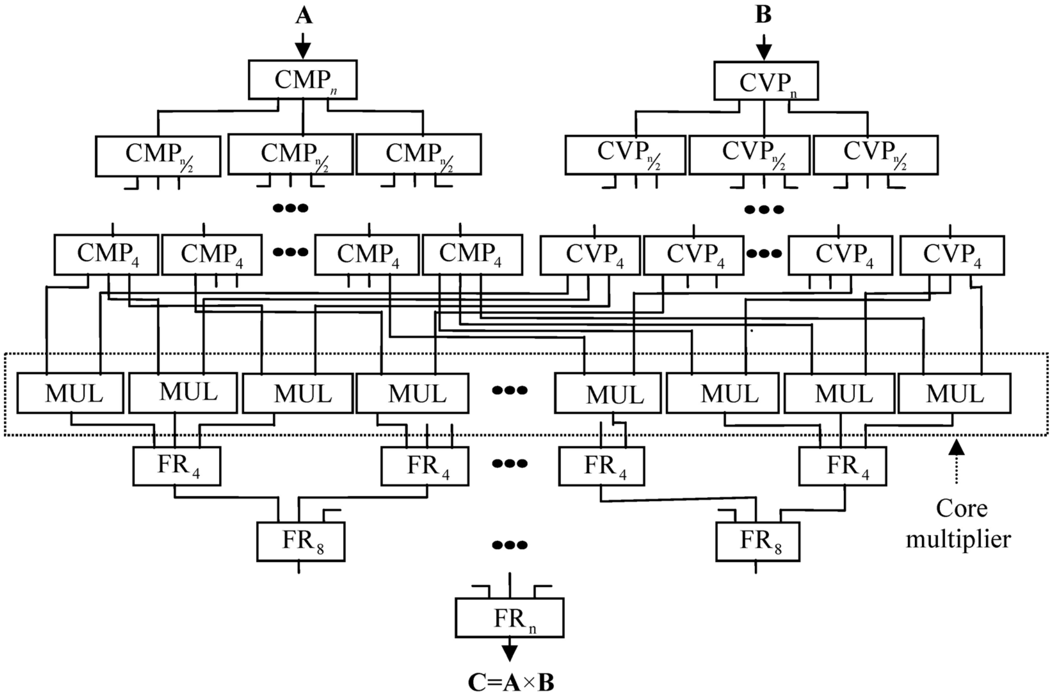


Fig. 7. The proposed subquadratic GNB multiplier

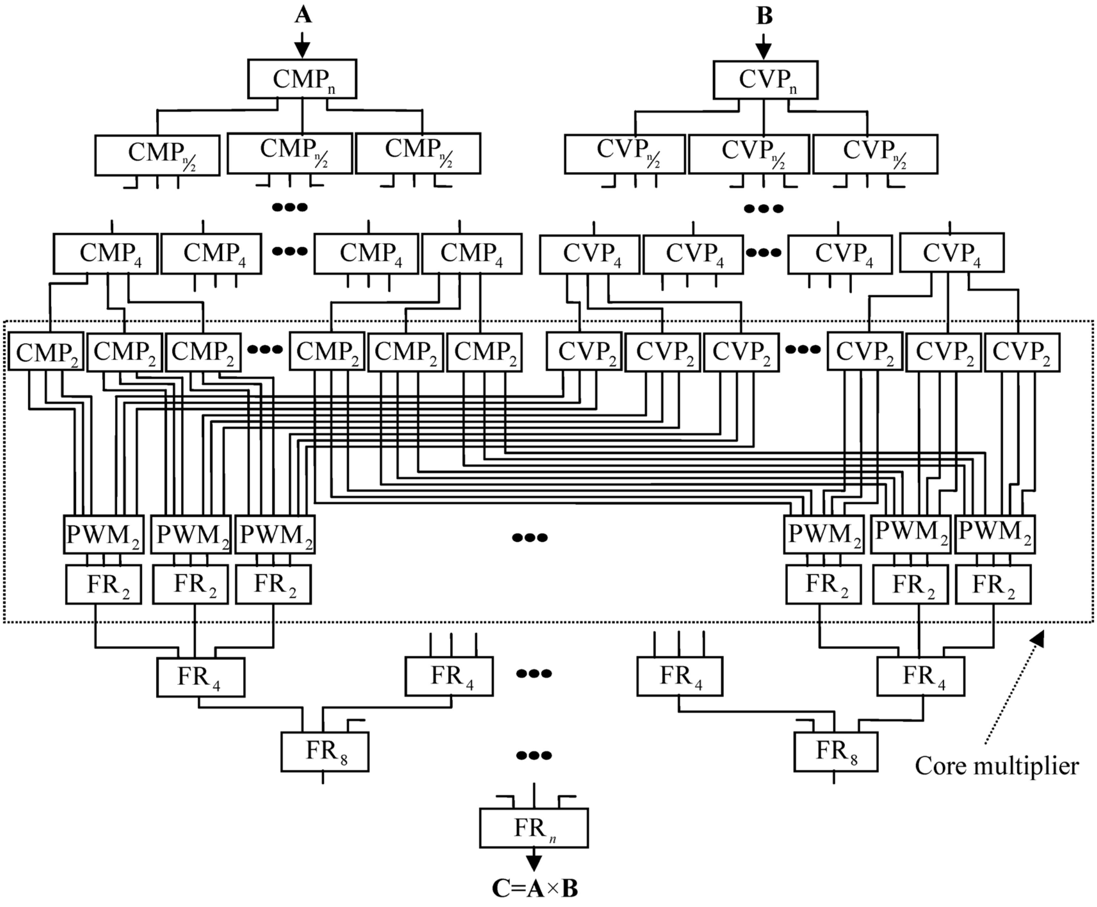


Fig. 8. The traditional TMVP GNB multiplier

Equation (6) is a $p \times p$ matrix with $1 \times p$ multiplication. Because p is a prime and odd number, one may first add one zero at the end of vectors **A** and **B** and extend the Toeplitz matrix from $p \times p$ matrix to $(p+1) \times (p+1)$ matrix through the insertion of zeros at elements *(0,p)* and *(p,0),* as follows:

$$\begin{bmatrix} c_{F(0)} \\ c_{F(1)} \\ c_{F(2)} \\ \cdots \\ c_{F(p-1)} \\ c_p \end{bmatrix} = \begin{bmatrix} a_{F(0)} & a_{F(1)} & a_{F(2)} & \cdots & a_{F(p-1)} & 0 \\ a_{F(p-1)} & a_{F(0)} & a_{F(1)} & \cdots & a_{F(p-2)} & a_{F(p-1)} \\ a_{F(p-2)} & a_{F(p-1)} & a_{F(0)} & \cdots & a_{F(p-3)} & a_{F(p-2)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{F(1)} & a_{F(2)} & a_{F(3)} & \cdots & a_{F(0)} & a_{F(1)} \\ 0 & a_{F(1)} & a_{F(2)} & \cdots & a_{F(p-1)} & a_{F(0)} \end{bmatrix} \times \begin{bmatrix} b_{F(0)} \\ b_{F(1)} \\ b_{F(2)} \\ \cdots \\ b_{F(p-1)} \\ 0 \end{bmatrix}. \qquad \textbf{(7)}$$

Following computation, the result bit c_p is disregarded. If the size of the Toeplitz matrix is odd, then Eq. (7) can be applied to make it even. Thus, the proposed multiplier can be used for computing $C=A \times B$.

4 Simplified Subquadratic GNB Multiplier

The proposed GNB multiplier in Fig. 7 can be further simplified on GNB multiplication itself. To overcome the difficulties of NB multiplication, we transform type-*t* GNB into a polynomial basis with $m \times t$ elements. In other words, this PB has *t* multiples of *m* elements in GNB. Each coefficient in GNB has *t* repeated elements in PB. After computing Eq. (6), the results $c_{F(0)}, c_{F(1)}, \ldots, c_{F(p-1)}$ consist of *t* multiples of the result $c_0, c_1, \ldots, c_{m-1}$. In fact, only one of the *t* multiples is actually required. Therefore, Eq. (6) can be simplified by reducing the order of matrix and vector. The algorithm used to find the minimum order q is described as follows:

```
Algorithm A: minimum-order-finding
/* Finding minimum order in Eq.(6) for GNB with type-t in GF($2^m$) */
/* p=mt+1                                                       */
/* q is the output minimum order of simplified Eq.(6)           */
Begin
For i=0 to m-1 do S[i]=p;
For i=0 to m-1 do
Begin
      For j=0 to t-1 do
        Begin
          temp=$2^{mj+i}$ mod p;
          If temp<S[i] then S[i]=temp;
        End;
    End;
q=S[0];
For i=1 to m-1 do If S[i]>q then q=S[i];
q=q+1;
End;
```

After computing Algorithm A, we obtain the minimum order q, such that Eq. (6) can be rewritten as follows:

$$\begin{bmatrix} c_{F(0)} \\ c_{F(1)} \\ c_{F(2)} \\ \cdots \\ c_{F(q-1)} \end{bmatrix} = \begin{bmatrix} a_{F(0)} & a_{F(1)} & a_{F(2)} & \cdots & a_{F(p-1)} \\ a_{F(p-1)} & a_{F(0)} & a_{F(1)} & \cdots & a_{F(p-2)} \\ a_{F(p-2)} & a_{F(p-1)} & a_{F(0)} & \cdots & a_{F(p-3)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{F(p-q+1)} & a_{F(p-q+2)} & a_{F(p-q+3)} & \cdots & a_{F(p-q)} \end{bmatrix} \times \begin{bmatrix} b_{F(0)} \\ b_{F(1)} \\ b_{F(2)} \\ \cdots \\ b_{F(p-1)} \end{bmatrix}. \tag{8}$$

Equation (8) can be divided and extended into two Toeplitz matrix-vector products as follows:

$$\begin{bmatrix} c_{F(0)} \\ c_{F(1)} \\ c_{F(2)} \\ \cdots \\ c_{F(q-1)} \end{bmatrix} = \begin{bmatrix} a_{F(0)} & a_{F(1)} & a_{F(2)} & \cdots & a_{F(q-1)} \\ a_{F(p-1)} & a_{F(0)} & a_{F(1)} & \cdots & a_{F(q-2)} \\ a_{F(p-2)} & a_{F(p-1)} & a_{F(0)} & \cdots & a_{F(q-3)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{F(p-q+1)} & a_{F(p-q+2)} & a_{F(p-q+3)} & \cdots & a_{F(0)} \end{bmatrix} \times \begin{bmatrix} b_{F(0)} \\ b_{F(1)} \\ b_{F(2)} \\ \cdots \\ b_{F(q-1)} \end{bmatrix} +$$

$$\begin{bmatrix} a_{F(q)} & \cdots & a_{F(p-1)} & 0 & 0 & 0 & \cdots & 0 \\ a_{F(q-1)} & \cdots & a_{F(p-2)} & a_{F(p-1)} & 0 & 0 & \cdots & 0 \\ a_{F(q-2)} & \cdots & a_{F(p-3)} & a_{F(p-2)} & a_{F(p-1)} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ a_{F(1)} & \cdots & a_{F(p-q)} & a_{F(p-q+1)} & a_{F(p-q+2)} & a_{F(p-q+3)} & \cdots & a_{F(q)} \end{bmatrix} \times \begin{bmatrix} b_{F(q)} \\ \cdots \\ b_{F(p-1)} \\ 0 \\ \cdots \\ 0 \end{bmatrix}. \tag{9}$$

We compute the two $q \times q$ Toeplitz matrix-vector products in Eq. (9). To reduce space complexity, only one $q \times q$ of the proposed subquadratic GNB multiplier is required for the computation of two $q \times q$ Toeplitz matrix products sequentially.

Example 1: Let *m*=13. One can find type-4 and p=53. By applying the algorithm of minimum-order-finding, one can obtain the minimum order of 23. Thus, the original 53×53 Toeplitz matrix could be reduced to a 23×53 Toeplitz matrix. Thus, the Toeplitz matrix-vector product based on Eq. (8) can be expressed as follows:

$$\begin{bmatrix} c_{F(0)} \\ c_{F(1)} \\ c_{F(2)} \\ \cdots \\ c_{F(22)} \end{bmatrix} = \begin{bmatrix} a_{F(0)} & a_{F(1)} & a_{F(2)} & \cdots & a_{F(52)} \\ a_{F(52)} & a_{F(0)} & a_{F(1)} & \cdots & a_{F(51)} \\ a_{F(51)} & a_{F(52)} & a_{F(0)} & \cdots & a_{F(50)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{F(31)} & a_{F(32)} & a_{F(33)} & \cdots & a_{F(0)} \end{bmatrix} \times \begin{bmatrix} b_{F(0)} \\ b_{F(1)} \\ b_{F(2)} \\ \cdots \\ b_{F(52)} \end{bmatrix}. \tag{10}$$

In accordance with Eq. (9), the above equation can be rewritten as follows:

$$\begin{bmatrix} c_{F(0)} \\ c_{F(1)} \\ c_{F(2)} \\ \cdots \\ c_{F(22)} \end{bmatrix} = \begin{bmatrix} a_{F(0)} & a_{F(1)} & a_{F(2)} & \cdots & a_{F(22)} \\ a_{F(52)} & a_{F(0)} & a_{F(1)} & \cdots & a_{F(21)} \\ a_{F(51)} & a_{F(52)} & a_{F(0)} & \cdots & a_{F(20)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{F(31)} & a_{F(32)} & a_{F(33)} & \cdots & a_{F(0)} \end{bmatrix} \times \begin{bmatrix} b_{F(0)} \\ b_{F(1)} \\ b_{F(2)} \\ \cdots \\ b_{F(22)} \end{bmatrix} + \begin{bmatrix} a_{F(23)} & \cdots & a_{F(52)} & 0 & 0 & 0 & \cdots & 0 \\ a_{F(22)} & \cdots & a_{F(51)} & a_{F(52)} & 0 & 0 & \cdots & 0 \\ a_{F(21)} & \cdots & a_{F(50)} & a_{F(51)} & a_{F(52)} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{F(1)} & \cdots & a_{F(30)} & a_{F(31)} & a_{F(32)} & a_{F(33)} & \cdots & a_{F(23)} \end{bmatrix} \times \begin{bmatrix} b_{F(23)} \\ \cdots \\ b_{F(52)} \\ 0 \\ \cdots \\ 0 \end{bmatrix}. \quad \textbf{(11)}$$

Thus, we obtain the simplified Toeplitz matrix-vector product for *m*=13.

Fig. 9 presents the proposed simplified subquadratic GNB multiplier. The dimension *p* in Fig. 7 is reduced to *q* in Fig. 9.

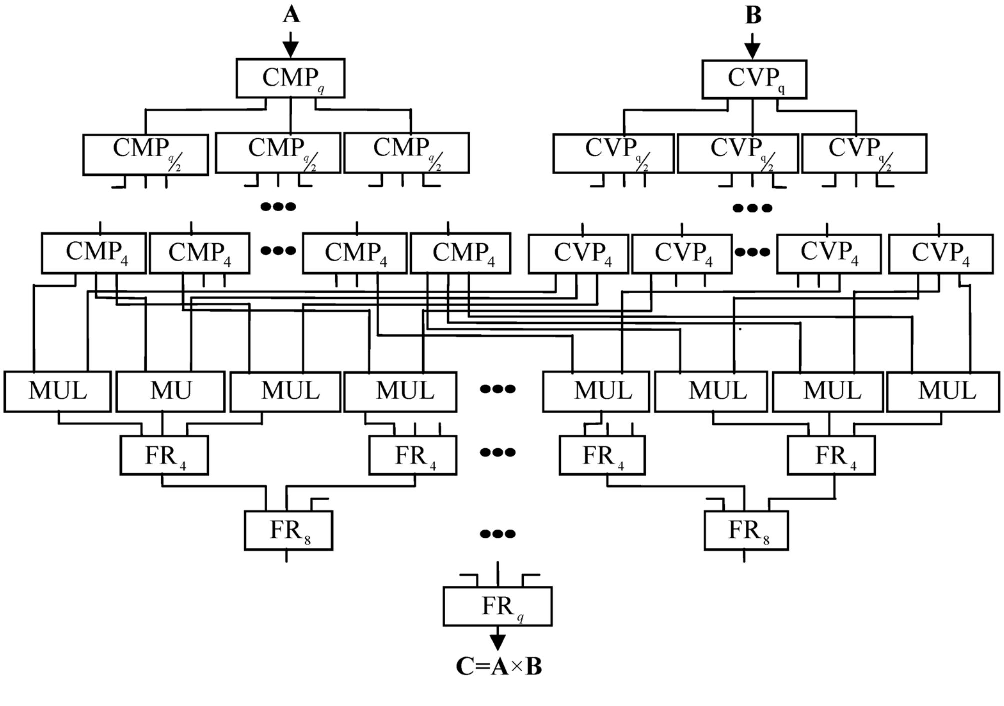


Fig. 9. The proposed simplified subquadratic GNB multiplier

5 Comparisons

Table 1 lists the space and time complexities of various subquadratic GNB multipliers. Fan and Hasan [28] proposed optimal type-1 and type-2 normal basis multipliers. To enable a fair comparison, we used their type-2 multiplier for comparison. Yang et al. [33] proposed a digit-serial subquadratic type-*t* GNB multiplier. We also compared the GNB multiplier in [33] with digit size *d*=*m* with our proposed multiplier. NanGate's Library Creator and the 45-nm FreePDK process kit [43] was used to synthesize the proposed multiplier. The cell areas of the 2-input AND gate and 2-input XOR gate are 1.064 μm^2 and 1.596 μm^2, respectively. In the case of input transition=0.0012ns and load capacitance=0.3656 fF, the propagation delays of a 2-input AND gate and a 2-input XOR gate are 0.02ns and 0.05ns, respectively. The multiplier proposed by Fan and Hasan [28] is type-2; therefore, we selected some m values with type-2 for comparison. Table 2 presents the comparison results. The proposed multiplier saves about 63% and 26% space complexities as compared to Fan-Hasan multiplier [28] and Yang multiplier [33], respectively. Moreover, the proposed multiplier saves 21% and 17% time complexities while comparing with Fan-Hasan multiplier [28] and Yang multiplier [33], respectively.

Table 1. Complexities of subquadratic GNB multipliers in GF(2^m)

Multipliers	Fan and Hasan [28]	Yang et al. [33]	The proposed multiplier (Fig. 9)
Basis	ONB with type-2	GNB with type-*t*	GNB with type-*t*
Structure	Bit-parallel	Digit-Serial (*d*=*m*)	Bit-parallel
		Space complexity	
#AND	$2m^{\log_2 3}$	$m^{\log_2 3}$	$\frac{4}{3} q^{\log_2 3}$
#XOR	$11m^{\log_2 3} - 12m + 1$	$5.5m^{\log_2 3} - 6m + 0.5$	$\frac{67}{18} q^{\log_2 3} - 6q + 0.5$
		Time complexity	
Time delay	$(2\log_2 m + 1)T_X + T_A$	$(2\log_2 m)\ T_X + T_A$	$(2\log_2 q - 3)T_X + T_A$

Table 2. Comparisons of subquadratic complexity GNB multipliers with type-2

Multipliers		Fan and Hasan [28]	Yang et al. [33]	The proposed multiplier (Fig. 9)
m	type-*t*		Space complexity (unit:μm^2)	
131	2	42151	21075	15443
233	2	106785	53392	39361
419	2	273960	101414	136980
593	2	477635	238817	177143
1013	2	1123194	561596	417486
The proposed multiplier saves space complexity as compared to Fan and Hasan [28] in average:				63%
The proposed multiplier saves space complexity as compared to Yang et al. [33] in average:				26%
m	type-*t*		Time complexity (unit: ns)	
131	2	0.7733423	0.723342	0.573342
233	2	0.856419	0.806419	0.656419
419	2	0.941081	0.891081	0.741081
593	2	0.991189	0.941189	0.791189
1013	2	1.068442	0.868442	1.018442
The proposed multiplier saves time complexity as compared to Fan and Hasan [28] in average:				21%
The proposed multiplier saves time complexity as compared to Yang et al. [33] in average:				17%

Let symbol S denotes space complexity. The symbols $S^{\oplus}(n)$ and $S^{\otimes}(n)$ stand for the number of 2-input XOR gates and 2-input AND gates, respectively. The space complexity of the proposed multiplier in Fig. 9 is computed as follows:

(a) The CMP requires the following $S^{\oplus}_{CMP}(q)$ 2-input XOR gates:

$$S^{\oplus}_{CMP}(q) = \begin{cases} S^{\oplus}_{CMP}(1) = 0 \\ S^{\oplus}_{CMP}(2) = 0 \\ S^{\oplus}_{CMP}(4) = 5 \\ 3 \times S^{\oplus}_{CMP}(q/2) + 3q/2 - 1 \end{cases} . \tag{12}$$

(b) The CVP needs the following $S^{\oplus}_{CVP}(q)$ 2-input XOR gates:

$$S^{\oplus}_{CVP}(q) = \begin{cases} S^{\oplus}_{CVP}(1) = 0 \\ S^{\oplus}_{CVP}(2) = 0 \\ S^{\oplus}_{CVP}(4) = 2 \\ 3 \times S^{\oplus}_{CVP}(q/2) + q/2 \end{cases} . \tag{13}$$

(c) The MUL consists of $2\times 3^{\log_2 q-2}$ 2-input XOR gates and $4\times 3^{\log_2 q-2}$ 2-input AND gates.

(d) The FR has the following $S_{FR}^{\oplus}(q)$ 2-input XOR gates:

$$S_{FR}^{\oplus}(q)=\begin{cases}S_{FR}^{\oplus}(1)=0\\ S_{FR}^{\oplus}(2)=0\\ S_{FR}^{\oplus}(4)=4\\ 3\times S_{FR}^{\oplus}(q/2)+q\end{cases}. \tag{14}$$

The space complexities of various similar multipliers are listed in Table 1.

6 Conclusions and Future Research

The TMVP approach has recently been applied to the derivation of multipliers over GF(2^m) with a subquadratic complexity computation architecture. This study also adopted the TMVP approach in the design of a bit-parallel subquadratic type-*t* GNB multiplier. However, the core multiplier in the proposed GNB multiplier uses a direct AND-XOR circuit structure rather than TMVP structure. The proposed core multiplier replaces the core CMP, CVP, PWM, and FR circuits found in existing TMVP multipliers (as shown in Fig. 5). Compared to the multiplier in [33], the proposed multiplier reduces space complexity by 26% and time complexity by 17%. The concept of this paper can be applied for other bases multipliers with subquadratic space complexity approach. In the future, we will present other bases multipliers with subquadratic and quadratic hybrid approach for achieving less space complexity. The optimized level sizes of subquadratic and quadratic should be also considered for the lowest space complexity in the future research.

Acknowledgments

The authors would like to thank anonymous referees and the editor for their carefully reading the paper and for their great help in improving the paper.

References

[1] V.S. Miller, Use of elliptic curves in cryptography, in: Proc. of Crypto 85, (LNCS, 218), 1986.

[2] N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation 48(1987) 203-209.

[3] D. Boneh, M.K. Franklin, Identity-based encryption from the weil pairing, SIAM Journal on Computing 32(3)(2003) 586-615.

[4] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21(1978) 120-126.

[5] IEEE Standard 1363-2000, IEEE standard specifications for public-key cryptography, January, 2000.

[6] ANSI X9.62-2005, Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), American National Standards Institute (ANSI), November, 2005.

[7] T.C. Bartee, D.J. Schneider, Computation with finite fields, Information and Computing 6(1963) 79-98.

[8] E.D. Mastrovito, VLSI architectures for multiplication over finite field GF(2^m), in: Proc. Sixth Int'l Conf. on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, (AAECC-6), 1988.

[9] Ç.K. Koç, B. Sunar, Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields, IEEE Trans. Computers 47(3)(1998) 353-356.

[10] T. Itoh, S. Tsujii, Structure of parallel multipliers for a class of fields GF(2^m), Information and Computation 83(1989) 21-40.

[11] C.-Y. Lee, C.-S. Yang, B.K. Meher, P.K. Meher, J.-S. Pan, Low-complexity digit-serial and scalable SPB/GPB multipliers over large binary extension fields using (b,2)-way Karatsuba decomposition, IEEE Trans. Circuits and Systems-I: Regular Papers 61(11)(2014) 3115-3124.

[12] H. Fan, M.A. Hasan, A new approach to subquadratic space complexity parallel multipliers for extended binary fields, IEEE Trans. Computers 56(2)(2007) 224-233.

[13] W.-T. Huang, C.H. Chang, C.W. Chiou, S.-Y. Tan, Non-XOR approach for low-cost bit-parallel polynomial basis multiplier over GF(2^m), IET Information Security 5(3)(2011) 152-162.

[14] J. Xie, J.J. He, P.K. Meher, Low latency systolic Montgomery multiplier for finite field GF(2^m) based on pentanomials, IEEE Trans. VLSI Systems 21(2)(2013) 385-389.

[15] C.-Y. Lee, P.K. Meher, W.-Y. Lee, Subquadratic space complexity digit-serial multiplier over binary extension fields using Toom-Cook algorithm, in: Proc. 2014 International Symposium on Integrated Circuits (ISIC), 2014.

[16] E.R. Berlekamp, Bit-serial reed-solomon encoder, IEEE Trans. Inf. Theory IT-28(1982) 869-874.

[17] H. Wu, M.A. Hasan, I.F. Blake, New low-complexity bit-parallel finite field multipliers using weakly dual bases, IEEE Trans. Computers 47(11)(1998) 1223-1234.

[18] M. Wang, I.F. Blake, Bit serial multiplication in finite fields, SIAM J. Disc. Math. 3(1)(1990) 140-148.

[19] J.-H. Wang, H.W. Chang, C.W. Chiou, W.-Y. Liang, Low-complexity design of bit-parallel dual basis multiplier over GF(2^m), IET Information Security 6(4)(2012) 324-328.

[20] Y.Y. Hua, J.-M. Lin, C.W. Chiou, C.-Y. Lee, Y.H. Liu, A novel digit-serial dual basis Karatsuba multiplier over GF(2^m), Journal of Computers 23(2)(2012) 80-94.

[21] J.-S. Pan, R. Azarderakhsh, M.M. Kermani, C.-Y. Lee, W.-Y. Lee, C.W. Chiou, J.-M. Lin, Low-latency digit-serial systolic double basis multiplier over GF(2^m) using subquadratic Toeplitz matrix-vector product approach, IEEE Trans. on Computers 63(5)(2014) 1169-1181.

[22] J.L. Massey, J.K. Omura, Computational method and apparatus for finite field arithmetic, U.S. Patent Number 4,587,627, May, 1986.

[23] C.C. Wang, T.K. Troung, H.M. Shao, L.J. Deutsch, J.K. Omura, I.S. Reed, VLSI architectures for computing multiplications and inverses in GF(2^m), IEEE Trans. Computers C-34(8)(1985) 709-717.

[24] A. Reyhani-Masoleh, Efficient algorithms and architectures for field multiplication using Gaussian normal bases, IEEE Trans. Computers 55(1)(2006) 34-47.

[25] G.B. Agnew, R.C. Mullin, I.M. Onyszchuk, S.A. Vanstone, An implementation for a fast public-key cryptosystem, Journal of Cryptology 3(1991) 63-79.

[26] M.A. Hasan, M.Z. Wang, V.K. Bhargava, A modified Massey-Omura parallel multiplier for a class of finite fields, IEEE Trans. Computers 42(10)(1993) 1278-1280.

[27] S. Kwon, A low complexity and a low latency bit parallel systolic multiplier over GF(2^m) using an optimal normal basis of type II, in: Proc. the 16th IEEE Symposium on Computer Arithmetic, Santiago de Compostela, 2003.

[28] H. Fan, M.A. Hasan, Subquadratic computational complexity schemes for extended binary field multiplication using optimal normal bases, IEEE Trans. Computers 56(10)(2007) 1435-1437.

[29] C.-Y. Lee, C.W. Chiou, Scalable Gaussian normal basis multipliers over GF(2^m) using Hankel matrix-vector representation, Journal of Signal Processing Systems for Signal Image and Video Technology 69(2)(2012) 197-211.

[30] C.W. Chiou, T.-P. Chuang, S.-S. Lin, C.-Y. Lee, J.-M. Lin, Y.-C. Yeh, Palindromic-like representation for Gaussian normal basis multiplier over GF(2^m) with odd type-t, IET Information Security 6(4)(2012) 318-323.

[31] C.W. Chiou, H.W. Chang, W.-Y. Liang, C.-Y. Lee, J.-M. Lin, Y.-C. Yeh, Low-complexity Gaussian normal basis multiplier over GF(2^m), IET Information Security 6(4)(2012) 310-317.

[32] R. Azarderakhsh, A. Reyhani-Masoleh, Low-complexity multiplier architectures for single and hybrid-double multiplications in Gaussian normal bases, IEEE Trans. Computers 62(4)(2013) 744-757.

[33] C.-S. Yang, J.-S. Pan, C.-Y. Lee, Digit-serial GNB multiplier based on TMVP approach over GF(2^m), in: Proc. 2013 Second International Conference on Robot, Vision and Signal Processing, 2013.

[34] C.W. Chiou, C.-C. Chang, C.-Y. Lee, T.-W. Hou, J.-M. Lin, Concurrent Error detection and Correction in Gaussian Normal Basis Multiplier over GF(2^m), IEEE Trans. Computers 58(6)(2009) 851-857.

[35] M. Leone, A new low complexity parallel multiplier for a class of finite fields, in: Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES 2001), LNCS 2162, 2001.

[36] D.W. Ash, I.F. Blake, S.A. Vanstone, Low complexity normal bases, Discrete Applied Math. 25(1989) 191-210.

[37] FIPS 186-2, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2, Nat'l Inst. of Standards and Technology, January, 2000.

[38] ISO/IEC 11770-3:2008, Information technology- Security techniques- Key management- Part 3: Mechanisms using asymmetric techniques, 2008.

[39] B. Sunar, A generalized method for constructing subquadratic complexity GF(2^k) multipliers, IEEE Trans. on Computers 53(9)(2004) 1097-1105.

[40] R.P. Brent, P. Gaudry, E. Thome, P. Zimmermann, Faster multiplication in GF(2) [*x*], in: ANTS-VIII 2008, LNCS 5011, 2008.

[41] J. Xie, C.-Y. Lee, P. K. Meher, Low-complexity systolic multiplier for GF(2^m) using Toeplitz matrix-vector product method, in: Proc. IEEE International Symposium on Circuits and Systems, Sapporo, Japan, 2019.

[42] J.-S. Pan, C.-Y. Lee, A. Sghaier, M. Zeghid, J. Xie, Novel systolization of subquadratic space complexity multipliers based on Toeplitz matrix-vector product approach, IEEE Trans. on Very Large Scale Integration (VLSI) Systems 27(7)(2019) 1614-1622.

[43] NanGate Standard Cell Library. <http://www.si2.org/openeda.si2.org/projects/nangatelib/>.