This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

New Architectures for Digit-Level Single, Hybrid-Double, Hybrid-Triple Field Multiplications and Exponentiation Using Gaussian Normal Bases

Hayssam El-Razouk and Arash Reyhani-Masoleh

***Abstract*—Gaussian normal bases (GNBs) are special set of normal bases (NBs) which yield low complexity $GF(2^m)$ arithmetic operations. In this paper, we present new architectures for the digit-level single, hybrid-double, and hybrid-triple multiplication of $GF(2^m)$ elements based on the GNB representation for odd values of $m > 1$. The proposed fully-serial-in single multipliers perform multiplication of two field elements and offer high throughput when the data-path capacity for entering inputs is limited. The proposed hybrid-double and hybrid-triple digit-level GNB multipliers perform, respectively, two and three field multiplications using the same latency required for a single digit-level multiplier, at the expense of increased area. In addition, we present a new eight-ary field exponentiation architecture which does not require precomputed or stored intermediate values.**

***Index Terms*—Digit-Level Multipliers, Finite Fields, Finite Field Exponentiation, Finite Field Multiplication, Gaussian Normal Basis, Hybrid-Double Multiplication, Normal Basis.**

I. Introduction

BINARY extension fields $GF(2^m)$ are heavily used in cryptography, error control coding, random number generation, and digital signal processing [1], [2], [3], [4]. A $GF(2^m)$ element is usually represented with respect to (w.r.t) a basis by a set of m coordinates from the binary field $GF(2)$. A basis of $GF(2^m)$ is a set of m linearly independent field elements [1], [5]. Polynomial bases (PBs) and normal bases (NBs) are two popular representations of $GF(2^m)$ elements [1], [5]. In these two representations, the field addition of two elements is accomplished by bit-wise $GF(2)$ additions (XOR). On the other hand, multiplication is more complicated than addition, and is a basic field operation for accomplishing more involved computations such as field inversion and exponentiation. The latter is a fundamental operation for the Diffie-Hellman key exchange algorithm [6] and is also used for other cryptographic applications such as random number generation [7], [3], [4]. Exponentiation is usually accomplished through iterations of squarings and multiplications [5]. Therefore, efficient $GF(2^m)$ multipliers with low output latencies are demanded for increasing throughput of such an important field operation.

This work has been supported in part by Natural Sciences and Engineering Council (NSERC) Alexander Graham Bell Canada Graduate Scholarships-Doctoral (CGS D) scholarship and Discovery and Discovery Accelerate Supplement (DAS) Grants.

H. El-Razouk and A. Reyhani-Masoleh are with the Department of Electrical and Computer Engineering, Western University, London, Canada, e-mail: helrazou@uwo.ca, areyhani@uwo.ca.

In the NB representation, squarings are implemented as cyclic shifts [5]. Therefore, NBs are considered advantageous for use in the hardware designs of binary extension fields' arithmetic [8] due to the free cost of squaring operations. In particular, Gaussian normal bases (GNBs) - a special subset of the NBs which offer field operations with smaller area and time overheads compared to the general NB - are often used for efficient hardware implementations of field multiplication, for example see the IEEE standard [5] and the National institute of standards and technology (NIST) standard [9]. In this paper, we focus on field multiplication based on the GNB representation for $GF(2^m)$ with odd values of m which include the five fields recommended by NIST for Elliptic curve digital signature algorithm (ECDSA) [9].

Massey and Omura [10] proposed the original scheme for multiplication of two field elements in the NB representation. For clarity of reference, in what follows, we refer to the multiplication of two field elements as single multiplication. After the work of Massey and Omura, a number of designs were proposed in an attempt to optimize the throughput and/or space complexities of the single NB multiplier [11], [12], [13], [8], [14], [15]. Generally, the different proposed designs can be divided into two categories of parallel and digit-level computations. For high throughput, the parallel implementation generates all the output bits of the single multiplication in one clock cycle [11], [12], [13], [8]. In $GF(2^m)$, this is achieved by m^2 two input AND gates and a number of two input XOR gates which is quadratic/sub-quadratic in m. To trade-off between space and throughput, digit-level single multipliers are deployed [11], [16], [15], [14], [17]. In digit-level single multiplication, the space complexity is traded-off with the number of required clock cycles in such a way that d, $1 \leq d < m$, bits are processed in parallel during each one of the $k = \lceil \frac{m}{d} \rceil$ clock cycles of computations. In this paper, we propose three new digit-level architectures for the single GNB multiplication, which follow different input/output order schemes, as stated next.

There are three schemes in terms of types of inputs and output for the digit-level (DL) multipliers: parallel-in-parallel-out (PIPO) [18], [15], [14], serial-in-parallel-out (SIPO) [19], [16], and parallel-in-serial-out (PISO) [10], [17], [20]. In the PIPO and PISO, both inputs are preloaded to the input registers in advance of computations. The output is generated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

in parallel after k iterations in the PIPO, while it is generated one digit at a time throughout the k iterations in the PISO. The output in the SIPO scheme is generated in parallel after k iterations. There are two variants of the SIPO scheme. The first variant (denoted as SIPO in this paper) requires the preloading of only one input in a single multiplier [19]. The other variant is denoted fully-serial-in-parallel-out (FSIPO) which does not require any preloading of the operands. This makes the FSIPO multipliers advantageous for achieving high throughput in applications where the data path capacity for inputs preloading is small and m is large. In this paper, we propose two new digit-level architectures for the FSIPO single GNB multiplication, in addition to an area efficient version of the MSD DL-PISO single GNB multiplier which was originally presented in [21].

By combining a DL-PISO and a DL-SIPO single GNB multipliers, a DL-PIPO hybrid-double GNB multiplier has been recently proposed in [22], which performs two field multiplications using the same latency required for a single field multiplication (i.e. k iterations). It is noted that the authors of [22] have shown the hybrid-double multiplier to be useful for applications where two dependent field multiplications are involved, such as double exponentiation. In this paper, we propose a new DL-SIPO hybrid-double GNB multiplier. Moreover, and for the first time in literature, we propose a DL-PIPO hybrid-triple GNB multiplier by combining our proposed DL-PISO single and DL-SIPO hybrid-double GNB multipliers. The proposed digit-level hybrid-triple multiplication scheme accomplishes three field multiplications using the latency required for a single digit-level multiplication. Furthermore, and based on the new hybrid-triple GNB multiplier, we propose an eight-ary field exponentiation architecture. Compared to the existing eight-ary schemes [23], [24], the proposed architecture offers almost the same latency while it does not require any precomputation or storage of the field element's odd powers which are less than 8.

In the following, we present the contributions of this work.

Contributions

The contributions of this paper are summarized in Figure 1. In this paper, we propose seven new digit-level architectures for the $GF(2^m)$ single, hybrid-double, and hybrid-triple multiplication, in addition to a new digit-level architecture for the $GF(2^m)$ eight-ary exponentiation, based on the GNB representation when m is odd. The contributions of this work are explained as follows:

- We propose two new architectures for MSD/LSD DL-FSIPO single GNB multipliers (Figures 2a and 3) to increase throughput in data-path constrained applications for large m. The MSD version is extended / optimized based on Feng's original bit-level (BL) FSIPO NB multiplier [16], while the LSD version is presented for the first time in literature. Moreover, we derive complexities for our proposed architectures, as there are no space / time complexities presented in [16]. In addition, we reduce the XOR count through applying sub-expression sharing to the multiplication by the normal element β.

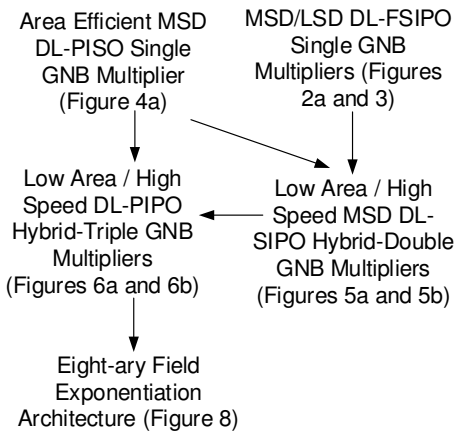


Figure 1: Summary of contributions.

- We propose an area efficient MSD DL-PISO single GNB multiplier (Figure 4a), where the number of XOR gates of the original MSD DL-PISO GNB multiplier in [21] is reduced by applying the sub-expression sharing presented in [25].
- We propose low area/high speed designs for an MSD DL-SIPO hybrid-double GNB multiplier (Figure 5). It is noted that the proposed digit-level hybrid-double GNB multiplier is the first SIPO scheme, as the one presented in [22] follows a DL-PIPO scheme.
- We propose, and for the first time in literature, low area/high speed designs for a DL-PIPO hybrid-triple GNB multiplier (Figure 6). The proposed digit-level PIPO hybrid-triple GNB multipliers perform three field multiplications using the latency of only one digit-level multiplication, at the expense of more space complexity.
- Finally, we propose an architecture which accomplishes eight-ary field exponentiation (Figure 8). The proposed scheme has almost same latency as offered by exiting eight-ary exponentiation schemes [23], [24], however, it does not require any precomputed or stored intermediate values.

The paper is organized as follows. In Section II, we give a brief introduction about multiplication in the GNB representation. In Section III, we present the proposed MSD/LSD DL-FSIPO single GNB multiplication schemes. Section IV explains the proposed MSD DL-PISO single GNB multiplier. Section V presents the proposed MSD DL-SIPO hybrid-double and DL-PIPO hybrid-triple GNB multiplication schemes. Section VI introduces the new eight-ary field exponentiation architecture. Section VII concludes the paper.

II. Preliminaries

In this section, we first briefly review formulations for the bit-level (BL) PISO multiplication of two $GF(2^m)$ elements represented in the GNB. After this, we show how one accomplishes the multiplication of an arbitrary field element by the normal element β.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

A. Formulation for the BL-PISO $GF(2^m)$ Single Multiplication in the GNB Representation

Here, we present the formulations for accomplishing bit-level PISO multiplication of two $GF(2^m)$ elements represented in the GNB.

For any $m > 1$ and not divisible by 8, if there exists a prime number $p = mT + 1$ such that $\gcd(mT/g, m) = 1$ where $2^g \equiv 1 \pmod p$, then, there exists a Gaussian normal basis (GNB) $\left\{\beta^{2^0}, \ldots, \beta^{2^{m-1}}\right\}$ of type T for representing the $GF(2^m)$ elements. Here, β is known as the normal element. It is noted that T is an even integer if m is odd. Any element $A \in GF(2^m)$ can be represented w.r.t the GNB as $A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = (a_0, \ldots, a_{m-1})$, where $a_i \in \{0, 1\}$.

Now, let $P_A(V) = AV = (p_0, \ldots, p_{m-1})$ denote the result of multiplying A by $V = (v_0, \ldots, v_{m-1})$. Then, by using the following formulation, one obtains the l-th coordinate of $P_A(V)$, for $0 \leq l < m$ [21]

$$p_l = a_l v_{((l+1))} + \sum_{i=1}^{m-1} a_{((l+i))} \left(\sum_{j=1}^{T} v_{((l+R[i,j]))} \right), \quad (1)$$

where $((q)) = q \bmod m$ and $0 \leq R[i,j] < m$, for $1 \leq i < m$ and $1 \leq j \leq T$, is an integer entry of an $(m-1) \times T$ matrix $\mathbf{R}$. $R[i,j]$ corresponds to the position of the j-th 1 in the i-th row of the GNB's multiplication matrix $\mathbf{M}$ [21]. This scheme for computing the l-th coordinate of the field multiplication requires m AND gates and at most $(m-1)T$ XOR gates, with a propagation delay of $T_A + (\lceil \log_2 m \rceil + \lceil \log_2 T \rceil) T_X$ [22], where T_A and T_X denote the propagation delay in a two input AND gate and a two input XOR gate, respectively.

In the following section, we show how the GNB multiplication of an arbitrary field element by the normal element β is accomplished.

B. Multiplication by the Normal Element β

Here, we present the formulation for accomplishing field multiplication of an arbitrary $GF(2^m)$ element $V = (v_0, \ldots, v_{m-1})$ represented in the Gaussian normal basis $\left\{\beta, \ldots, \beta^{2^{m-1}}\right\}$ of type T by the normal element $\beta = (1, 0, \ldots, 0)$. By substituting for $(a_0, \ldots, a_{m-1})$ with $(1, 0, \ldots, 0)$ in (1), and considering all values of $l = 0, \ldots, m-1$, we obtain [21]

$$P_\beta(V) = v_1 \beta + \sum_{i=1}^{m-1} \left(\sum_{j=1}^{T} v_{((i+R[m-i,j]))} \right) \beta^{2^i}, \quad (2)$$

which requires at most $(m-1)(T-1)$ XOR gates, with a propagation delay of $\lceil \log_2 T \rceil T_X$. It is noted that for $T > 2$, one can reduce the number of XOR gates required for realizing (1) or (2) through applying signal reuse to $\mathbf{M}$ (see [26], [27] for example), where the amount of XOR savings is obtained through simulation.

In the following section, we present the formulations and architectures for the proposed MSD / LSD DL-FSIPO single GNB multipliers. We also derive the formulations for space and time complexities of these proposed multipliers.

III. Proposed DL-FSIPO Single GNB Multipliers

In the following, we start by presenting the proposed MSD DL-FSIPO single GNB multiplier, followed by the LSD one. In addition to their proofs, the proposed digit-level formulations presented in this section, i.e. (3), (4), (5), and (6), have been verified through simulations using the Sage tool [28]. It is noted that the proposed multipliers in this section do not require preloading of inputs, and perform the multiplication operation as the input digits enter the multiplier. This is advantageous, especially for large values of m, to achieve high throughput in applications where the parallel preloading of the inputs is not possible due to limited capacity of the data path.

A. Proposed MSD DL-FSIPO Single GNB Multiplier

In this section, we propose a digit-level MSD architecture for the FSIPO single GNB multiplication. In what follows, we first derive the formulations for the MSD DL-FSIPO single multiplication in the GNB. This is followed by presenting the proposed architecture of the MSD DL-FSIPO single GNB multiplier. The section ends by analyzing the space and time complexities.

1) Formulations: In this section, we derive formulations for digit-level multiplication of two $GF(2^m)$ elements represented in the GNB, where the two inputs of the multiplier are entered serially, digit-by-digit, in an MSD first order. In what follows, we show the proposed MSD first recursive construction of field elements when represented in the GNB.

Lemma 1. *Given a digit size $0 < d < m$, we construct a field element $A = (a_0, \ldots, a_{m-1}) \in GF(2^m)$ represented in the GNB, recursively, starting from the most significant digit A_{k-1} (total of $k = \lceil \frac{m}{d} \rceil$ digits A_0 through A_{k-1}), as follows:*

$$A^{(i)} = A_{k-1-i} + \left(A^{(i-1)}\right)^{2^d} \quad (3)$$

where i takes values from 0 to $k-1$, $A^{(-1)} = 0$, $A = A^{(k-1)}$, and $A_{k-1-i} = \sum_{j=0}^{d-1} a_{d(k-1-i)+j} \beta^{2^j}$ is the $(k-1-i)$-th digit of $A = (A_0, \ldots, A_{k-1})$ with $a_{d(k-1-i)+j} = 0$ for $d(k-1-i)+j \geq m$.

Proof. By substituting for $i = 0, \ldots, k-1$ in (3), we get

$$\begin{aligned} A^{(k-1)} &= A_0 + \left(A_1 + \cdots \left(A_{k-2} + (A_{k-1})^{2^d} \right)^{2^d} \cdots \right)^{2^d} \\ &= \sum_{i=k-1}^{0} A_{k-1-i}^{2^{d(k-1-i)}}, \end{aligned}$$

and by noticing that $A_{k-1-i} = \sum_{j=0}^{d-1} a_{d(k-1-i)+j} \beta^{2^j}$ we obtain

$$\begin{aligned} A^{(k-1)} &= \sum_{i=k-1}^{0} \sum_{j=0}^{d-1} a_{j+d(k-1-i)} \beta^{2^{j+d(k-1-i)}} \\ &= \sum_{j=0}^{d-1} a_j \beta^{2^j} + \sum_{j=0}^{d-1} a_{j+d} \beta^{2^{j+d}} + \cdots + \\ &\quad \sum_{j=0}^{d-1} a_{j+d(k-1)} \beta^{2^{j+d(k-1)}} = \sum_{j=0}^{m-1} a_j \beta^{2^j}, \end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

where the last result is achieved since $a_{j+d(k-1)} = 0$ for $j + d(k-1) \geq m$. □

Then, the multiplication of the $GF(2^m)$ elements A and B is obtained as follows.

Proposition 1. *Let $E = AB$ be the multiplication of the two elements $A, B \in GF(2^m)$ represented in the GNB. By using construction* (3)*, one obtains $E = A^{(k-1)}B^{(k-1)}$ through the following recurrence proceeding from $i = 0$ to $k-1$*

$$A^{(i)}B^{(i)} = \sum_{j=0}^{d-1} \left(\left(a_{d(k-1-i)+j} \left(B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d} \right) + b_{d(k-1-i)+j} \left(A^{(i-1)} \right)^{2^d} \right)^{2^{-j}} \beta \right)^{2^j} + \left(A^{(i-1)}B^{(i-1)} \right)^{2^d}. \tag{4}$$

Proof. $A^{(i)}B^{(i)}$ is obtained by substituting for $A^{(i)}$ and $B^{(i)}$ in $A^{(i)}B^{(i)}$, using (3), as

$$\begin{aligned} A^{(i)}B^{(i)} &= \left(A_{k-1-i} + \left(A^{(i-1)}\right)^{2^d} \right) \left(B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d} \right) \\ &= A_{k-1-i} \left(B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d} \right) + B_{k-1-i} \left(A^{(i-1)} \right)^{2^d} + \left(A^{(i-1)}B^{(i-1)} \right)^{2^d}, \end{aligned}$$

and by substituting for $A_{k-1-i} = \sum_{j=0}^{d-1} a_{d(k-1-i)+j}\beta^{2^j}$ in $A_{k-1-i}\left(B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d} \right)$, and for $B_{k-1-i} = \sum_{j=0}^{d-1} b_{d(k-1-i)+j}\beta^{2^j}$ in $B_{k-1-i}\left(A^{(i-1)}\right)^{2^d}$ we get

$$A^{(i)}B^{(i)} = \sum_{j=0}^{d-1} a_{d(k-1-i)+j}\beta^{2^j} \left(B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d} \right) + \sum_{j=0}^{d-1} b_{d(k-1-i)+j}\beta^{2^j} \left(A^{(i-1)} \right)^{2^d} + \left(A^{(i-1)}B^{(i-1)} \right)^{2^d}$$

which yields

$$A^{(i)}B^{(i)} = \sum_{j=0}^{d-1} \left(\left(a_{d(k-1-i)+j} \left(B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d} \right) + b_{d(k-1-i)+j} \left(A^{(i-1)} \right)^{2^d} \right)^{2^{-j}} \beta \right)^{2^j} + \left(A^{(i-1)}B^{(i-1)} \right)^{2^d}.$$

□

It is noted that the correctness of (4) has also been verified using simulations with the Sage tool [28]. It is also noted that Lemma 1 and Proposition 1 are applicable to general NBs.

In (4), the multiplication of A by B (elements of $GF(2^m)$) represented in the GNB, is reduced recursively to a number of bit-wise AND operations, field additions, multiplications with the normal element β, and cyclic shifts for computing the powers 2^{-j}, 2^j, and 2^d. Notice that the addition of the digit B_{k-1-i} to $\left(B^{(i-1)}\right)^{2^d}$ in (4) is a free of cost concatenation. This is because the most significant digit of $B^{(i-1)}$ is 0^d for $0 \leq i < k$, where 0^d denotes a string of zeros of length d.

Since it is already given that $A^{(-1)} = B^{(-1)} = 0$, therefore by using (4), starting at $i = 0$, and proceeding up to $i = k-1$ (k clock cycles), we obtain the final result of the multiplication $E = A^{(k-1)}B^{(k-1)}$. At each step, the $(k-1-i)$-th digit of A and B, i.e. A_{k-1-i} and B_{k-1-i}, in addition to $A^{(i-1)}$, $B^{(i-1)}$, and $A^{(i-1)}B^{(i-1)}$, are used for computing $A^{(i)}$ and $B^{(i)}$, and $A^{(i)}B^{(i)}$ according to (3) and (4), respectively.

During the review phase of this paper, it has been pointed out that the proposed multiplication algorithm in Propositions 1 and 2 (see Section III-B1) look similar to the one presented in [29]. In fact, Propositions 1 and 2 build a $GF(2^m)$ element recursively digit-by-digit, starting from the MSD and LSD, respectively. This behavior results in a DL-FSIPO GNB multiplication scheme. The algorithm presented in [29] takes opposite action by recursively shrinking a $GF(2^m)$ element bit-by-bit, starting from the LSB. The latter behavior constructs a BL-PIPO GNB multiplication scheme, but not a DL-FSIPO GNB one. In addition, the authors of [29] present a bit-parallel GNB multiplier extended from their algorithm. Bit-parallel multipliers do not require any input or output registers for their processing and usually target high throughput applications by generating the output in one clock cycle at the expense of a space complexity which is quadratic in m for the scheme in [29]. On the other hand, this paper focuses on digit-level multiplications for resource constrained applications which requires input / output registers and trade-off space complexity against larger number of clock cycles.

Next, we present the proposed architecture of the MSD DL-FSIPO single GNB multiplier.

2) Architecture: Figure 2 presents the architecture of the proposed MSD DL-FSIPO single GNB multiplier. In this

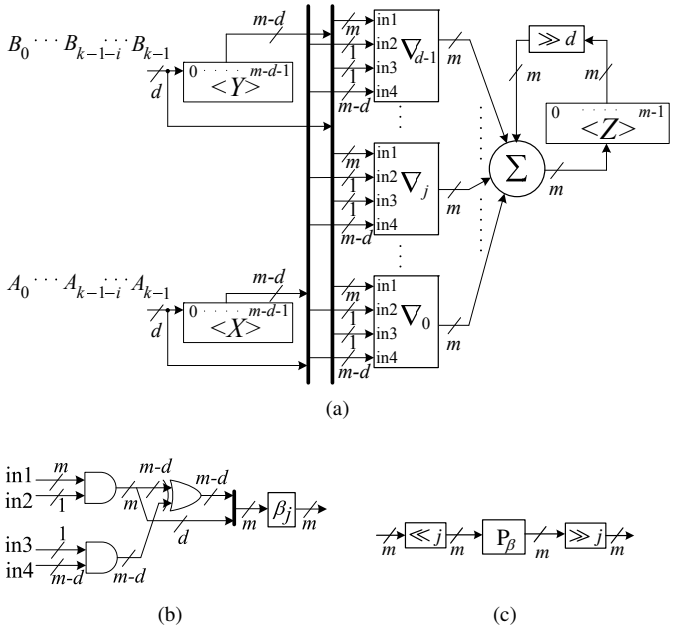


Figure 2: (a) Architecture of the proposed MSD DL-FSIPO single GNB multiplier. (b) Architecture of ∇_j. (c) Architecture of β_j.

figure, for $0 \leq j < d$ and $0 \leq i < k$, we have: in1 $= B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d}$, in2 $= a_{d(k-1-i)+j}$, in3 $= b_{d(k-1-i)+j}$, and in4 $= \left(A^{(i-1)}\right)^{2^d}$. $\ll j$ and $\gg j$, respectively, represent left and right j-bits cyclic shifts. P_β represents the multiplication

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

by β. This architecture is constructed based on (3) and (4). In Figure 2a, d denotes digit size, $k = \lceil \frac{m}{d} \rceil$ denotes total number of cycles of computations, and $0 \leq i < k$ refers to i-th clock cycle.

Initially, the $(m-d)$-bits shift registers $\langle X \rangle$ and $\langle Y \rangle$, and the m-bits register $\langle Z \rangle$, are cleared (i.e., initialized by $A^{(-1)}$, $B^{(-1)}$, and $A^{(-1)}B^{(-1)}$, respectively). Then, at each i-th iteration of the following k iterations, $\langle X \rangle$, $\langle Y \rangle$, and $\langle Z \rangle$ update their states from $A^{(i-1)}$, $B^{(i-1)}$, and $A^{(i-1)}B^{(i-1)}$ to $A^{(i)}$, $B^{(i)}$, and $A^{(i)}B^{(i)}$, respectively, as follows. The two $GF(2^m)$ input elements A and B are entered to registers $\langle X \rangle$ and $\langle Y \rangle$, respectively, one digit per a clock cycle, following a most significant digit first order starting with the $(k-1)$-th digits (according to (3)). At the i-th iteration, $\langle X \rangle$ and $\langle Y \rangle$ perform a d-fold right shift (not cyclic) and, the $(k-1-i)$-th digits of A and B are written to the least significant d-bits of $\langle X \rangle$ and $\langle Y \rangle$, respectively. At the same time, register $\langle Z \rangle$ accumulates the result of the field addition $\sum_{j=0}^{d-1} \nabla_j + \left(A^{(i-1)}B^{(i-1)}\right)^{2^d}$, where

$$\nabla_j = \Bigg(\Bigg(a_{d(k-1-i)+j} \left(B_{k-1-i} + \left(B^{(i-1)} \right)^{2^d} \right) + b_{d(k-1-i)+j} \left(A^{(i-1)} \right)^{2^d} \Bigg)^{2^{-j}} \beta \Bigg)^{2^j}$$

is generated as shown in Figures 2b and 2c. According to (4), this results in writing $A^{(i)}B^{(i)}$ to $\langle Z \rangle$. Then, after k clock cycles, i.e. $i = k-1$, we obtain $\langle Z \rangle = A^{(k-1)}B^{(k-1)} = AB$. It is noted that the proposed architecture implements $B_{k-1-i} + \left(B^{(i-1)}\right)^{2^d}$ in (4) by concatenating the d-bits of B_{k-1-i} to the least significant digit of $\left(B^{(i-1)}\right)^{2^d}$ (the concatenations are shown by thick vertical lines in Figure 2, two in Figure 2a and one in Figure 2b). This concatenation is possible since the least significant digit of $\left(B^{(i-1)}\right)^{2^d}$ is simply 0^d for all $0 \leq i < k$ (notice from (4) that $A^{(k-1)}$ and $B^{(k-1)}$ are not used in generating $A^{(k-1)}B^{(k-1)}$).

It is worth mentioning that, although our presented DL-FSIPO multiplication algorithms are different from the one in [29], however, they meet at the bit-parallel level. Accordingly, one might construct a multiplexer based DL-FSIPO GNB multipliers through applying partitioning to the bit-parallel architecture presented in [29] (our proposed DL-FSIPO single GNB multipliers are AND / XOR based). In this case, similar efforts to those presented in this paper need to be taken in order to optimize number of FFs and number of XOR gates within the fixed multiplication by β. Also, notice that, the underlying multiplication algorithm needs to be theoretically aligned / proved according to our formulations. Otherwise, it would be more natural to construct a DL-PIPO GNB multiplier by partitioning of the architecture in [29], which reflects the underlying multiplication algorithm presented in [29]. In fact, the missing of reference to Feng's original work [16] throughout [29] indicates that authors of [29] were determined to use a DL-PIPO algorithm.

In the following, we study the space and time complexities of the proposed MSD DL-FSIPO single GNB multiplier.

3) Space and Time Complexities: The space complexity of the proposed MSD DL-FSIPO single GNB multiplier is listed in Table I. This includes the count of logic gates, Flip Flop (FF), and preloading multiplexers. In this table, T is the GNB type, while P and S denote either the corresponding input/output is loaded/generated in parallel or in serial, respectively. It is noted that this table shows the space complexity of the proposed MSD DL-FSIPO single GNB multiplier before applying sub-expression sharing. From Figure 2b, one can see that each ∇_j module, $0 \leq j < d$, consists of $m+m-d = 2m-d$ two input AND gates, and therefore, the total number of two input AND gates in the d ∇_j modules of Figure 2a is $d(2m-d)$. The total number of two input XOR gates in the $\sum$ module of Figure 2a (a $GF(2^m)$ adder which adds $d+1$ field elements) is dm. In addition, each ∇_j module has $\leq (m-d)+(T-1)(m-1)$ XORs out of which $\leq (T-1)(m-1)$ are contributed by the multiplications by β (before sub-expression elimination, see Section II-B). Therefore, the total number of XORs in the MSD DL-FSIPO single GNB multiplier is $\leq d[(2m-d)+(T-1)(m-1)]$. In addition, while register $\langle Z \rangle$ has m FFs, only $m-d$ FFs are required for each of registers $\langle X \rangle$ and $\langle Y \rangle$, since the $(k-1)$-th digits in these two registers are always zeros throughout the computations. Hence, the total number of FFs is $2(m-d)+m = 3m-2d$. One can also see that there are no preloading multiplexers required for the proposed MSD DL-FSIPO single GNB multiplier.

On the other hand, Table II reports the time complexity of the proposed digit-level MSD FSIPO single GNB multiplier, in terms of the propagation delay of the corresponding levels of two input XOR and AND gates through the critical path. As seen from Figure 2a, the critical path of the proposed architecture passes through one ∇_j module and the $\sum$ module. The propagation delay of a ∇_j module is $T_A + (1 + \lceil \log_2(T) \rceil) T_X$, where $\lceil \log_2(T) \rceil T_X$ is the propagation delay through β_j (due to the multiplication with β, see Section II-B). Therefore, by adding the delay of the $\sum$ module (a $GF(2^m)$ adder which adds $d+1$ field elements), which is $\lceil \log_2(d+1) \rceil T_X$, the total propagation delay of the proposed multiplier becomes $T_A + [1 + \lceil \log_2(d+1) \rceil + \lceil \log_2 T \rceil] T_X$.

4) Bit-Level Case: It is noted that the original bit-level FSIPO NB multiplication scheme was presented by Feng [16] for an MSB order of the inputs. By considering a single-bit digit size, one obtains a bit-level MSB FSIPO single GNB multiplier from our proposed digit-level architecture. Our MSB BL-FSIPO single GNB multiplier offers a maximum propagation delay of $T_A + 3T_X$, while it requires 13 FFs, 9 ANDs, and 13 XORs (for $GF(2^5)$ and $T = 2$). On the other hand, the $GF(2^5)$ multiplier presented in [16] has a maximum propagation delay of $T_A + 6T_X$ and requires 13 FFs, 9 ANDs, and 15 XORs. In addition, we further reduce the space complexity of our proposed architecture (for $T > 2$) through applying sub-expression sharing techniques to the multiplication by β (see [26], [27] for example).

Table III estimates the corresponding space and time complexity readings for the case of bit-level ($d = 1$) versions of the different multipliers in Tables I and II, considering the type-4 GNB of $GF(2^{163})$, based on the STMicroelectronics 65nm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

Table I: Space complexity of digit-level single GNB multipliers. PIL / SIL = Parallel / Serial input loading.

Multiplier	FF	AND	XOR	2 : 1 1-bit MUX		Input 1	Input 2	Output
				PIL	SIL			
DL-PISO [10]	$2m$	$d\left[T\left(m-1\right)+1\right]$	$d\left[T\left(m-1\right)\right]$	$2m$	$2d$	P	P	S
DL-PISO [17]	$2m$	dm	$d\left[T\left(m-1\right)\right]$	$2m$	$2d$	P	P	S
DL-PISO[1] [22]	$2m$	dm	$\leq d\left[\left(T-1\right)\left(\left(m-1\right)-\frac{d-1}{2}\right)\right]+d\left(m-1\right)$	$2m$	$2d$	P	P	S
DL-PIPO [25]	$3m$	dm	$d\left[\frac{(m-1)(T-1)}{2}+m\right]$	$2m$	$2d$	P	P	P
DL-SIPO[1] [22]	$2m$	dm	$\leq d\left(T-1\right)\left[\left(m-1\right)-\frac{d-1}{2}\right]+dm$	m	d	S	P	P
MSD / LSD DL-FSIPO[2] (Figures 2 & 3)	$3m-2d$	$d\left(2m-d\right)$	$\leq d\left[\left(2m-d\right)+\left(T-1\right)\left(m-1\right)\right]$	0	0	S	S	P
MSD DL-PISO[1] (Figure 4a)	$2m$	dm	$\leq d\left[\left(T-1\right)\left(\left(m-1\right)-\frac{d-1}{2}\right)\right]+d\left(m-1\right)$	$2m$	$2d$	P	P	S

[1] without applying group sub-expression elimination. [2] without applying sub-expression elimination.

Table II: Time complexity of digit-level single GNB multipliers in terms of number of two input AND and XOR gates levels.

Multiplier	Propagation Delay	Serial Preloading Latency	Computation Latency
DL-PISO [10]	$T_A+\left(\lceil\log_2\left(T\left(m-1\right)+1\right)\rceil\right)T_X$	k	k
DL-PISO [17]	$T_A+\left(\lceil\log_2 m\rceil+\lceil\log_2 T\rceil\right)T_X$	k	k
DL-PISO [22]	$T_A+\left(\lceil\log_2 m\rceil+\lceil\log_2 T\rceil\right)T_X$	k	k
DL-PIPO [25]	$T_A+\left(\lceil\log_2\left(d+1\right)\rceil+\lceil\log_2 T\rceil\right)T_X$	k	k
DL-SIPO [22]	$T_A+\left(\lceil\log_2\left(d+1\right)\rceil+\lceil\log_2 T\rceil\right)T_X$	k	k
MSD / LSD DL-FSIPO (Figures 2 & 3)	$T_A+\left[1+\lceil\log_2\left(d+1\right)\rceil+\lceil\log_2 T\rceil\right]T_X$	0	k
MSD DL-PISO (Figure 4a)	$T_A+\left(\lceil\log_2 m\rceil+\lceil\log_2 T\rceil\right)T_X$	k	k

Table III: Space and time complexity readings for the case of type-4 GNB of $GF\left(2^{163}\right)$ bit-level single multipliers. TP/G estimated in Kbps/Gate.

Multiplier	CPD	Serial Input Loading			Parallel Input Loading		
	nsec	Total Gates[2]	Latency	TP/G @ 1 GHz	Total Gates[2]	Latency	TP/G @ 1 GHz
BL-PISO [10]	0.43	3333.75	326	150	3981.75	164	250
BL-PISO [17]	0.43	2726.25	326	183	3374.25	164	295
BL-PIPO [25]	0.15	2853.5	326	175	3501.5	164	284
LSB BL-SIPO [22]	0.15	2726.25	326	183	3050.25	164	326
MSB / LSB BL-FSIPO[1] (Figures 2 & 3, $d=1$)	0.19	3854.5	163	259	3854.5	163	259

[1] without elimination. If we apply the elimination in [26], savings is 127 XORs = 254 GE. [2] with MUXs.

CMOS standard library. Synthesis results based on this library using Synopsys Design Vision tool reports the area (measured in $(\mu m)^2$) for a two input NAND, two input AND, two input XOR, D-type FF, and a $2:1$ 1-bit MUX as 2.08, 2.6, 4.16, 7.8, and 4.16, respectively. The total NAND gate equivalency (GE) for any design is obtained through dividing its total area by the area of a single NAND gate. According to the same library, the propagation delay measured in nano seconds (nsec) for a two input NAND, two input AND, two input XOR, D-type FF, and a $2:1$ 1-bit MUX are 0.02, 0.03, 0.04, 0.05, and 0.03, respectively. In Table III, CPD denotes critical path delay. Latency denotes the number of clock cycles required for computing the m-bits of output. TP is throughput (@ 1 GHz) and TP/G denotes throughput per total GE estimated in Kbps/Gate. Throughput is estimated as the number of the multiplier's output bits (i.e., m) divided by latency and multiplied by a speed of 1 GHz, that is, $TP=\frac{m}{latency}\times(1\text{ GHz})$. As one can see from this table, the bit-level version of the proposed DL-FSIPO single GNB multiplier offers half the latency and provides the best normalized throughput compared to the other multipliers in the case of serial loading of inputs. Moreover, one can further reduce the space complexity of the proposed architecture through applying sub-expression sharing techniques to the multiplication by β. For example, if we apply the elimination algorithm proposed in [26], we save 127 XORs which is equivalent to 254 GE.

Note that, even though a DL-SIPO single GNB multiplier requires only one parallel input, the problem of preloading of this single input might still exist if m is large and the underlying application is resource constrained and has a limited capacity input data-path (for example, NIST recommends $163\leq m\leq 571$ for ECDSA [9]).

In the following section, we introduce the proposed LSD DL-FSIPO single GNB multiplier.

B. Proposed LSD DL-FSIPO Single GNB Multiplier

In this section, we present the LSD DL-FSIPO single GNB multiplier. We start by deriving the formulations for the LSD DL-FSIPO single GNB multiplication scheme. Then we present the architecture of the proposed multiplier. We end this section by studying space and time complexities.

1) Formulations: Here, we derive the formulations for digit-level multiplication of two $GF\left(2^m\right)$ elements based on the GNB representation, where the two inputs of the multiplier

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

are entered in an LSD first order. In the following we first show how we construct the elements of $GF(2^m)$ when represented in the GNB, digit by digit, starting from the least significant digit.

Lemma 2. *Given the digit size $0 < d < m$, an arbitrary $GF(2^m)$ element $A = (a_0, \ldots, a_{m-1})$ represented in the GNB is constructed, recursively, starting with its least significant digit ($k = \lceil \frac{m}{d} \rceil$ digits), as follows:*

$$A^{(i)} = \left(A_i + A^{(i-1)}\right)^{2^{-d}} \tag{5}$$

where i takes values from 0 to $k-1$, $A = A^{(k-1)}$, $A^{(-1)} = 0$, and $A_i = \sum_{j=0}^{d-1} a_{di+j-r}\beta^{2^j}$ is the i-th digit of A such that $a_{di+j-r} = 0$ for $di + j - r < 0$ given $r = kd - m$ which represents the number of left padded zeros.

Proof. By substituting for $i = 0, \ldots, k-1$ in (5), we get

$$\begin{aligned} A^{(k-1)} &= \left(A_{k-1} + \cdots \left(A_1 + (A_0)^{2^{-d}}\right)^{2^{-d}} \cdots\right)^{2^{-d}} \\ &= \sum_{i=0}^{k-1} A_i^{2^{-d(k-i)}}, \end{aligned}$$

and by noticing that $A_i = \sum_{j=0}^{d-1} a_{di+j-r}\beta^{2^j}$ we obtain

$$\begin{aligned} A^{(k-1)} &= \sum_{i=0}^{k-1}\sum_{j=0}^{d-1} a_{di+j-r}\beta^{2^{j-d(k-i)}} \\ &= \left(\sum_{i=0}^{k-1}\sum_{j=0}^{d-1} a_{di+j-r}\beta^{2^{j+di}}\right)^{2^{-dk}}. \end{aligned}$$

Now, let $l = di + j$, and notice that $dk = m + r$ (since $r = kd - m$), then

$$\begin{aligned} A^{(k-1)} &= \left(\sum_{l=0}^{m+r-1} a_{l-r}\beta^{2^l}\right)^{2^{-m-r}} \\ &= \left(\sum_{l=-r}^{m-1} a_l\beta^{2^{l+r}}\right)^{2^{-r}} = \sum_{l=0}^{m-1} a_l\beta^{2^l}, \end{aligned}$$

where the last result is achieved since $a_l = 0$ for $l < 0$. □

Then, the multiplication of two $GF(2^m)$ elements A and B represented in the GNB and constructed by (5), is obtained as follows.

Proposition 2. *Let $E = AB$ be the multiplication of two elements $A, B \in GF(2^m)$ represented in the GNB. Therefore, using construction* (5)*, $E = A^{(k-1)}B^{(k-1)}$ is obtained by the following recurrence*

$$\begin{aligned} A^{(i)}B^{(i)} = \Bigg[\sum_{j=0}^{d-1}\Bigg(\Bigg(a_{di+j-r}\left(B_i + B^{(i-1)}\right) + \\ b_{di+j-r}A^{(i-1)}\Bigg)^{2^{-j}}\beta\Bigg)^{2^j} + A^{(i-1)}B^{(i-1)}\Bigg]^{2^{-d}}, \end{aligned} \tag{6}$$

where i proceeds from 0 to $k-1$, and $A^{(-1)} = B^{(-1)} = A^{(-1)}B^{(-1)} = 0$.

Proof. $A^{(i)}B^{(i)}$ is obtained by substituting for $A^{(i)}$ and $B^{(i)}$ in $A^{(i)}B^{(i)}$, using (5), as

$$\begin{aligned} A^{(i)}B^{(i)} &= \left[\left(A_i + A^{(i-1)}\right)\left(B_i + B^{(i-1)}\right)\right]^{2^{-d}} \\ &= \left[A_i\left(B_i + B^{(i-1)}\right) + B_iA^{(i-1)} + A^{(i-1)}B^{(i-1)}\right]^{2^{-d}}, \end{aligned}$$

and by substituting for $A_i = \sum_{j=0}^{d-1} a_{di+j-r}\beta^{2^j}$ in $A_i\left(B_i + B^{(i-1)}\right)$, and for $B_i = \sum_{j=0}^{d-1} b_{di+j-r}\beta^{2^j}$ in $B_iA^{(i-1)}$ we get

$$\begin{aligned} A^{(i)}B^{(i)} = \Bigg[&\sum_{j=0}^{d-1} a_{di+j-r}\beta^{2^j}\left(B_i + B^{(i-1)}\right) + \\ &\sum_{j=0}^{d-1} b_{di+j-r}\beta^{2^j}A^{(i-1)} + A^{(i-1)}B^{(i-1)}\Bigg]^{2^{-d}} \end{aligned}$$

which yields

$$\begin{aligned} A^{(i)}B^{(i)} = \Bigg[\sum_{j=0}^{d-1}\Bigg(\Bigg(&a_{di+j-r}\left(B_i + B^{(i-1)}\right) + \\ &b_{di+j-r}A^{(i-1)}\Bigg)^{2^{-j}}\beta\Bigg)^{2^j} + A^{(i-1)}B^{(i-1)}\Bigg]^{2^{-d}}. \end{aligned}$$

□

Notice that Lemma 2 and Proposition 2 are also applicable to general NBs. In (6), and similar to (4), the multiplication of A by B (elements of $GF(2^m)$) represented in the GNB, is reduced recursively to a number of bit-wise AND operations, field additions, multiplications with the normal element β, and cyclic shifts for computing the powers 2^{-j}, 2^j, and 2^{-d}. It is noted that the field addition of the term B_i in (6) is realized through concatenation. The concatenation is possible since the least significant digit of $B^{(i-1)}$ is always 0 for $0 \leq i < k$ (only $B^{(k-1)}$ has a non zero LSD; however, $B^{(k-1)}$ is not used in computing $A^{(k-1)}B^{(k-1)}$).

Since it is already given that $A^{(-1)} = B^{(-1)} = 0$, therefore, starting at $i = 0$, and proceeding up to $i = k-1$, we obtain the final result of the multiplication $AB = A^{(k-1)}B^{(k-1)}$. As one can see from (6), at each step, the i-th digits in A and B, i.e. A_i and B_i, together with $A^{(i-1)}$, $B^{(i-1)}$, and $A^{(i-1)}B^{(i-1)}$, are used for computing $A^{(i)}B^{(i)}$.

Next, we present the proposed architecture of the LSD DL-FSIPO single GNB multiplier.

2) Architecture: Here, we introduce the architecture of the proposed LSD DL-FSIPO single GNB multiplier. This architecture is shown in Figure 3, which is constructed based on (5) and (6). In this figure, i denotes the i-th clock cycle of the computations, $0 \leq i < k$. First, the $(m-d)$-bits shift registers $\langle X \rangle$ and $\langle Y \rangle$, and the m-bits register $\langle Z \rangle$ are cleared (in other words, $\langle X \rangle$, $\langle Y \rangle$, and $\langle Z \rangle$ are loaded with $A^{(-1)}$, $B^{(-1)}$, and $A^{(-1)}B^{(-1)}$, respectively). After this, and at each of the following i-th iterations, $0 \leq i < k$, registers $\langle X \rangle$, $\langle Y \rangle$, and $\langle Z \rangle$ change states from $A^{(i-1)}$, $B^{(i-1)}$, and $A^{(i-1)}B^{(i-1)}$ to $A^{(i)}$, $B^{(i)}$, and $A^{(i)}B^{(i)}$, respectively, as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

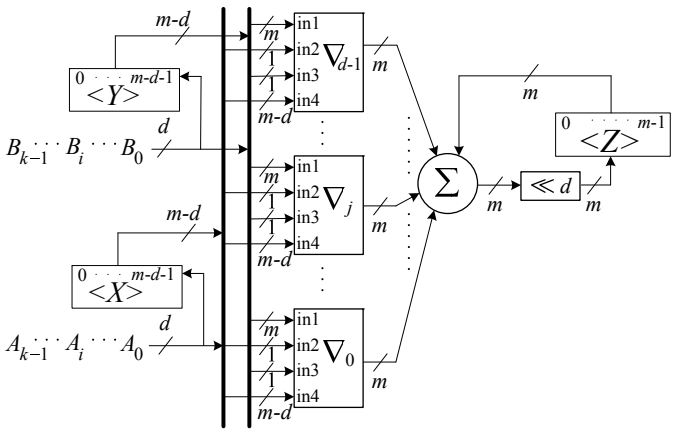


Figure 3: Architecture of the proposed LSD DL-FSIPO single GNB multiplier. Architecture of ∇_j and β_j (which is a component of ∇_j) blocks are shown in Figures 2b and 2c, respectively, where in Figure 2b, and at iteration $0 \leq i < k$, we have: in1 $= B_i + B^{(i-1)}$, in2 $= a_{di+j-r}$, in3 $= b_{di+j-r}$, and in4 $= A^{(i-1)}$.

follows. At iteration i, $\langle X \rangle$ and $\langle Y \rangle$ perform a d-fold left shift (not cyclic) and, at the same time, the i-th digits of the field elements A and B are written to the most significant d-bits of $\langle X \rangle$ and $\langle Y \rangle$, respectively, according to (5). Moreover, and at the same time, register $\langle Z \rangle$ accumulates the d-fold left cyclic shift of $\sum_{j=0}^{d-1} \nabla_j + A^{(i-1)} B^{(i-1)}$, where the architecture of ∇_j is captured in Figure 2b and implements

$$\left(\left(a_{di+j-r} \left(B_i + B^{(i-1)} \right) + b_{di+j-r} A^{(i-1)} \right)^{2^{-j}} \beta \right)^{2^j}$$

for in1 $= B_i + B^{(i-1)}$, in2 $= a_{di+j-r}$, in3 $= b_{di+j-r}$, and in4 $= A^{(i-1)}$. Therefore, after the i-th clock cycle, $\langle Z \rangle = A^{(i)} B^{(i)}$, as one can see from (6). After k clock cycles we get $\langle Z \rangle = A^{(k-1)} B^{(k-1)} = AB$. It is noted that, the least significant digit of $B^{(i-1)}$ is always 0^d for $0 \leq i < k$, where 0^d denotes a string of zeros of length d. Therefore, the proposed architecture implements $B_i + B^{(i-1)}$ in (6) by concatenating B_i to the least significant digit of $B^{(i-1)}$.

In the following, we analyze the space and time complexities of the LSD DL-FSIPO single GNB multiplier.

3) Space and Time Complexities: The space complexity of the proposed LSD DL-FSIPO single GNB multiplier is listed in Table I, in terms of the count of logic gates, FFs, and preloading multiplexers. In Section III-A3, we found that each ∇_j module, $0 \leq j < d$, has $2m - d$ AND gates and $\leq (m-d) + (T-1)(m-1)$ XOR gates. Figure 3, on the other hand, shows that the number of two input XOR gates in the $\sum$ module (a $GF(2^m)$ adder which adds $d+1$ field elements) is dm. And hence, the total number of two input AND gates is $d(2m-d)$, while the total number of XOR gates adds up to $\leq d[(2m-d) + (T-1)(m-1)]$. In addition, one can notice that, while register $\langle Z \rangle$ has m FFs, registers $\langle X \rangle$ and $\langle Y \rangle$ have $m-d$ FFs each, since their least significant digits will always be zero throughout the k clock cycles of computations. This adds up to a total of $3m - 2d$ FFs. It is also noted that no preloading MUXs required for the proposed LSD DL-FSIPO single GNB multiplier.

The time complexity of the proposed LSD DL-FSIPO single GNB multiplier is reported in Table II, in terms of levels of the propagation delay of two input XOR and AND gates through its critical path. Similar to the proposed MSD DL-FSIPO single GNB multiplier (see Section III-A3), Figure 3 shows that the propagation delay of the proposed LSD architecture is equivalent to the sum of the propagation delays through one ∇_j module and the $\sum$ module. Hence, the maximum propagation delay in the proposed LSD DL-FSIPO single GNB multiplier is $T_A + [1 + \lceil \log_2 (d+1) \rceil + \lceil \log_2 T \rceil] T_X$, as shown in Table II.

IV. Proposed DL-PISO Single GNB Multiplier

In this section, we present the proposed architecture of the area efficient MSD DL-PISO single GNB multiplier. This multiplier is an area-optimized instance of the one presented by the authors of [21], which is based on (1), and hence, the reader is referred to [21] for more details about the formulations. The area reduction is accomplished through applying the group sub-expression sharing algorithm presented in [25]. In what follows, we first show the architecture of the proposed multiplier, followed by analyzing its space and time complexities.

A. Architecture

Here, we present an area efficient architecture for the MSD DL-PISO single GNB multiplier, as it is shown in Figure 4a. Notice that, the proposed DL-PISO GNB multiplier requires

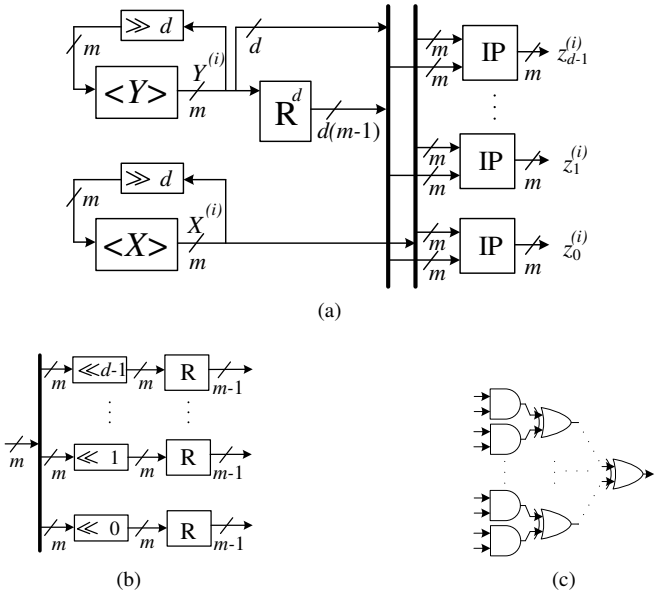


Figure 4: (a) The proposed architecture of the MSD DL-PISO single GNB multiplier. (b) Architecture of the R^d block before applying sub-expression sharing. (c) Architecture of the IP block. $0 \leq i < k$ denotes the iteration number, $k = \lceil \frac{m}{d} \rceil$.

additional m-bit $2:1$ multiplexers for parallel loading of inputs, while it requires additional d-bit $2:1$ multiplexers for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

serial loading of inputs. These multiplexers are not shown in Figure 4a. The architecture in this figure differs from the LSD DL-PISO single GNB multiplier, which is presented in [22], in that it generates the multiplication output in the order of most significant digit first. This is accomplished through generating the d output bits $z_{d-1}^{(i)}$ through $z_0^{(i)}$, during iteration i, where $0 \leq i < k$, as follows. In Figure 4a, a bit $z_n^{(i)}$ denotes the left most (least significant) coordinate of $P_{X^{(i)}}^{2^{-n}}\left(Y^{(i)}\right)$, where $0 \leq n < d$ and , $X^{(i)} = A^{2^{(i+1)d-t}}$ and $Y^{(i)} = B^{2^{(i+1)d-t}}$ ($t = k \times d - m$) denote the contents of registers $\langle X \rangle$ and $\langle Y \rangle$ at the i-th iteration of the computations. It is noted that $z_n^{(i)}$, the left most coordinate of $P_{X^{(i)}}^{2^{-n}}\left(Y^{(i)}\right)$, is obtained according to (1) as follows

$$z_n^{(i)} = x_n^{(i)} y_{n+1}^{(i)} + \sum_{u=1}^{m-1} x_{((n+u))}^{(i)} \left(\sum_{v=1}^{T} y_{((n+R[u,v]))}^{(i)} \right). \quad (7)$$

In (7), $x_j^{(i)}$ and $y_j^{(i)}$, respectively, denote the j-th coordinates (cells) of registers $\langle X \rangle$ and $\langle Y \rangle$ during the i-th iteration. Therefore, by initializing registers $\langle X \rangle$ and $\langle Y \rangle$ such that $X^{(0)} = A^{2^{d-t}}$ and $Y^{(0)} = B^{2^{d-t}}$, one obtains the most significant digit of the output. It is noted that the t-fold left cyclic shift in $A^{2^{d-t}}$ and $B^{2^{d-t}}$ is implemented in order to allow for appending zeros to the t most significant bits of the first output digit (most significant digit), i.e., $\left(z_{d-t}^{(0)}, \ldots, z_{d-1}^{(0)}\right)$. This is required for compatibility of integration with the proposed MSD DL-FSIPO single GNB multiplier (see Section V). Then, at the i-th iteration, $0 \leq i < k$, one has $X^{(i)} = A^{2^{(i+1)d-t}}$ and $Y^{(i)} = B^{2^{(i+1)d-t}}$, due to the d-fold right cyclic shifts which are applied to $\langle X \rangle$ and $\langle Y \rangle$ at each clock cycle, as shown in Figure 4a. According to this, bit $z_n^{(i)}$ of the i-th output digit in Figure 4a maps to the output bit $e_{m-(d(i+1)-t)+n}$ of the multiplication result $E = AB$. For all $0 \leq n < d$, the inner product in (7) is realized through an IP block in Figure 4a, while the d instances (for $0 \leq n < d$) of the $m-1$ bits of $\sum_{u=1}^{m-1} \left(\sum_{v=1}^{T} y_{((n+R[u,v]))} \right) \beta^{2^u}$ are generated through the R^d block, which is shown in Figure 4b. This figure shows the architecture of R^d before applying the group sub-expression sharing algorithm presented in [25], where each R block represents the matrix multiplication of the lower $m-1$ rows of the multiplication matrix $\mathbf{M}$ by the corresponding m-bits input vertical vector.

The following is the space and time complexity analysis of the proposed MSD DL-PISO single GNB multiplier.

B. Space and Time Complexities

Here, we discuss the space and time complexities of the proposed MSD DL-PISO single GNB multiplier in Figure 4a. In this figure, an IP block consists of m AND gates and $m-1$ XOR gates, as it is shown in Figure 4c. Furthermore, for area efficiency, the R^d block of Figure 4a realizes a group sub-expression shared version of the d R blocks in Figure 4b based on the algorithm presented in [25]. It is noted that one can find the number of eliminations due to this sharing through simulation. Therefore, the architecture of Figure 4a requires a total of $2m$ FFs, dm ANDs, and $\leq d\left[(T-1)\left((m-1)-\frac{d-1}{2}\right)\right] + d(m-1)$ XORs, as presented in Table I, where $\leq (T-1)(m-1)$ is the number of XOR gates in each R block before sub-expression sharing (see Section II-B) and the term $\frac{d(d-1)}{2}$ is due to the elimination of common rows between different $\mathbf{R}$ matrices which are used in implementing the d R blocks [22].

For the time complexity, one can see that the critical path of Figure 4a has a propagation delay equals to $T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$, as presented in Table I, where the delay through the area optimized R^d block is $\lceil \log_2 T \rceil T_X$ [22], and that through an IP module is $T_A + \lceil \log_2 m \rceil T_X$.

In the following, we combine the proposed MSD DL-FSIPO and DL-PISO single GNB multipliers to construct an MSD DL-SIPO hybrid-double and a DL-PIPO hybrid-triple GNB multipliers.

V. Proposed Digit-Level Hybrid-Double and Hybrid-Triple GNB Multipliers

A hybrid-double digit-level GNB multiplication architecture has been recently proposed by the authors of [22], which performs two field multiplications (multiplication of three field elements) using the same latency required for a single field multiplication (i.e. $k = \lceil \frac{m}{d} \rceil$ iterations for a digit size d). To accomplish this, the authors of [22] have extended the LSB BL-PISO GNB multiplier in [21] and the LSB BL-SIPO GNB multiplier in [19] to the digit-level, then, by combining these two digit-level single GNB multipliers, they constructed their DL-PIPO hybrid-double GNB multiplier.

In this section, we present four new architectures for $GF(2^m)$ digit-level hybrid multiplications, two for an MSD DL-SIPO hybrid-double multiplication (a low area and a high speed designs) and, for the first time, two architectures (low area / high speed) for a DL-PIPO hybrid-triple multiplication (multiplication of four field elements), when the field elements are represented in the GNB. In order to construct the proposed hybrid-double multiplier, we combine the MSD DL-PISO single GNB multiplier presented in Section IV-A with the proposed MSD DL-FSIPO single GNB multiplier of Figure 2a. On the other hand, the proposed hybrid-triple multiplier is constructed by combining the MSD DL-PISO single GNB multiplier presented in Section IV-A with the MSD DL-SIPO hybrid-double GNB multiplier proposed in this section[1].

In the following, we first present the proposed architectures of the MSD DL-SIPO hybrid-double GNB multiplier, followed by those of the proposed DL-PIPO hybrid-triple GNB multiplier. We conclude this section by analyzing the space and time complexities of the two proposed hybrid multipliers.

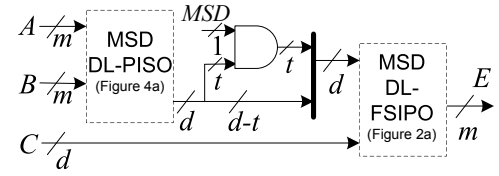
A. Proposed MSD DL-SIPO Hybrid-Double GNB Multiplier

This section presents the architecture of the proposed MSD DL-SIPO hybrid-double GNB multiplier. It is noted that the hybrid-double GNB multiplier proposed by the authors of [22] follows a DL-PIPO scheme of its inputs/outputs, while our

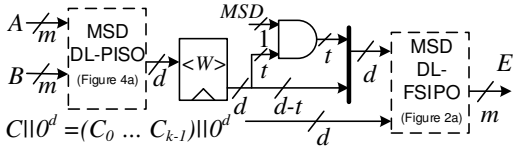
[1] It is noted that one can build an LSD DL-SIPO hybrid-double architecture, as well as a DL-PIPO hybrid-triple architecture, by combining the LSD DL-PISO GNB multiplier presented in [22] with the LSD DL-FSIPO GNB multiplier which is proposed in this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

proposed architecture is the first DL-SIPO scheme for the digit-level hybrid-double GNB multiplication. Figure 5 shows two versions of the proposed MSD DL-SIPO hybrid-double GNB multiplier, one for a low area design (Figure 5a) and the other for a high speed design (Figure 5b). It is noted that the



(a)



(b)

Figure 5: Architectures of the proposed MSD DL-SIPO hybrid-double GNB multiplier, where $t = k \times d - m$: (a) Low area design. (b) High speed design. The appended 0^d (zero digit) in input C balances the timing due to pipelining.

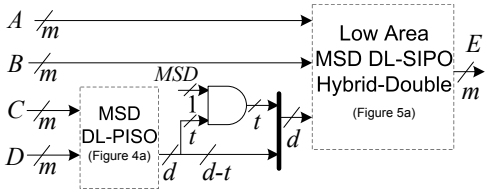
most significant t-bits - where $t = k \times d - m$, $k = \lceil \frac{m}{d} \rceil$, and d is the digit size - of the first output digit (most significant digit) of the MSD DL-PISO single GNB multipliers in Figure 5 are set to zero through the MSD signal. As can be seen from the figure, the low area MSD DL-SIPO hybrid-double GNB multiplier is built from MSD DL-PISO and DL-FSIPO single GNB multipliers with the output of the former connected to one input of the latter. On the other hand, the high speed version follows from the low area version by inserting the d-bits register $\langle W \rangle$ between the output of the DL-PISO, and the input of the DL-FSIPO, single GNB multipliers, as can be seen from Figure 5b. This, in turn, shortens the propagation delay of the multiplier's critical path, which results in reaching higher operating frequencies compared to the low area version. However, it adds one extra clock cycle to the latency. Each one of the two versions of the proposed MSD DL-SIPO hybrid-double GNB multiplier takes three inputs, two of which are m-bits wide each (inputs A and B in Figure 5), while the third one has only d-bits (input C in Figure 5).

Initially, A and B are loaded to the input registers of the MSD DL-PISO single GNB multiplier, while the input/output registers of the MSD DL-FSIPO single GNB multiplier are cleared out. In the low area version, at the i-th clock cycle, $0 \leq i < k$, the MSD DL-PISO single GNB multiplier generates the $(k-1-i)$-th output digit for the multiplication AB, while the MSD DL-FSIPO single GNB multiplier generates $E^{(i)} = (AB)^{(i)} C^{(i)}$ (according to (3) and (4)). After k iterations, the output register of the MSD DL-FSIPO single GNB multiplier holds the result of the triple multiplication, i.e., $E^{(k-1)} = (AB)^{(k-1)} C^{(k-1)}$. In the high speed version, an extra clock cycle is required at the beginning to store the MSD output digit of the DL-PISO single GNB multiplier to register $\langle W \rangle$.

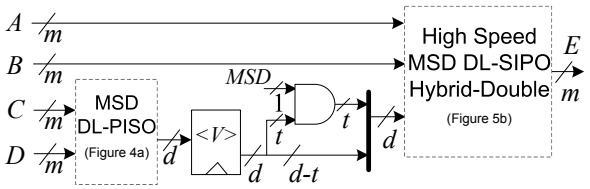
In the following, we introduce the proposed architectures for the DL-PIPO hybrid-triple GNB multiplier.

B. Proposed DL-PIPO Hybrid-Triple GNB Multiplier

In this section, we present the proposed architectures for the DL-PIPO hybrid-triple GNB multiplier. To the best of the authors' knowledge, this is the first digit-level hybrid GNB multiplier proposed in the literature which performs three $GF(2^m)$ multiplications using the same latency of a single digit-level field multiplication (multiplying of four field elements of A, B, C, and D together). Figures 6a and 6b present two variants of the proposed DL-PIPO hybrid-triple GNB multiplier. Figure 6a is a low area design, while Figure 6b shows a high speed design. The most significant t-bits,



(a)



(b)

Figure 6: Architectures of the proposed DL-PIPO hybrid-triple GNB multiplier: (a) Low area design. (b) High speed design.

$t = k \times d - m$, of the first output digit of the MSD DL-PISO single GNB multipliers in these figures are set to zero through the MSD signal. In Figure 6a, the low area DL-PIPO hybrid-triple GNB multiplier is constructed from one MSD DL-PISO single, and one low area MSD DL-SIPO hybrid-double, GNB multipliers with the output of the former connected to the serial input of the latter. The high speed DL-PIPO hybrid-triple GNB multiplier instance uses a high speed MSD DL-SIPO hybrid-double GNB multiplier. Also, it has a d-bits register $\langle V \rangle$ inserted between the output of the MSD DL-PISO single GNB multiplier and the input of the high speed MSD DL-SIPO hybrid-double GNB multiplier (see Figure 6b). This leads to shorter critical path, and hence, results in reaching higher operating frequencies compared to the low area instance; however, at the expense of one extra clock cycle.

Each one of the two proposed hybrid-triple GNB multiplier's versions takes four m-bits inputs, denoted by A, B, C, and D, and generates an m-bits output, i.e. $E = ABCD$. Initially, A, B, C, and D are loaded to the input registers of the MSD DL-PISO single GNB multipliers (including the one in the DL-SIPO hybrid-double GNB multiplier), while

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

Table IV: Space complexity of the digit-level hybrid-double and hybrid-triple GNB multipliers. PIL = Parallel input loading, SIL = Serial input loading.

Multiplier	D-FF	AND	XOR[1]	2 : 1 MUX	
				PIL	SIL
DL-PIPO Hybrid-Double[2] (low area) [22]	$4m$	$2dm+t$	$\leq d(T-1)[2(m-1)-(d-1)]+d(2m-1)$	$3m$	$3d$
DL-PIPO Hybrid-Double[2] (high speed) [22]	$4m+d$	$2dm+t$	$\leq d(T-1)[2(m-1)-(d-1)]+d(2m-1)$	$3m$	$3d$
MSD DL-SIPO Hybrid-Double (low area) (Figure 5a)	$5m-2d$	$d(3m-d)+t$	$\leq d(T-1)\left\lceil 2(m-1)-\frac{d-1}{2}\right\rceil+d(3m-(d+1))$	$2m$	$2d$
MSD DL-SIPO Hybrid-Double (high speed) (Figure 5b)	$5m-d$	$d(3m-d)+t$	$\leq d(T-1)\left\lceil 2(m-1)-\frac{d-1}{2}\right\rceil+d(3m-(d+1))$	$2m$	$2d$
DL-PIPO Hybrid-Triple (low area) (Figure 6a)	$7m-2d$	$d(4m-d)+2t$	$\leq d(T-1)[3(m-1)-(d-1)]+d(4m-(d+2))$	$4m$	$4d$
DL-PIPO Hybrid-Triple (high speed) (Figure 6b)	$7m$	$d(4m-d)+2t$	$\leq d(T-1)[3(m-1)-(d-1)]+d(4m-(d+2))$	$4m$	$4d$

[1] without sub-expression elimination. [2]Note: the authors of [22] did not count for the t ANDs which are required for appending zeros .

Table V: Time complexity of the digit-level hybrid-double and hybrid-triple GNB multipliers.

Multiplier	Propagation Delay	Serial Loading of Inputs Latency	Computation Latency
DL-PIPO Hybrid-Double (low area) [22]	$T_{PISO}+T_{SIPO}$	k	k
DL-PIPO Hybrid-Double (high speed) [22]	$\max\{T_{PISO},T_{SIPO}\}$	k	$k+1$
MSD DL-SIPO Hybrid-Double (low area) (Figure 5a)	$T_{PISO}+T_{FSIPO}+T_A$	k	k
MSD DL-SIPO Hybrid-Double (high speed) (Figure 5b)	$\max\{T_{PISO},T_{FSIPO}+T_A\}$	k	$k+1$
DL-PIPO Hybrid-Triple (low area) (Figure 6a)	$T_{PISO}+T_{FSIPO}+T_A$	k	k
DL-PIPO Hybrid-Triple (high speed) (Figure 6b)	$\max\{T_{PISO},T_{FSIPO}+T_A\}$	k	$k+1$

the input/output registers of the MSD DL-FSIPO single GNB multiplier (in the hybrid-double multiplier) are cleared out. In the low area version, at the i-th clock cycle, $0 \leq i < k$, the MSD DL-PISO single GNB multipliers generate their $(k-1-i)$-th output digits for AB and CD at the same time, while the output register of the MSD DL-SIPO hybrid-double multiplier computes $E^{(i)} = (AB)^{(i)}(CD)^{(i)}$ (according to (3) and (4)). After k iterations, the output register of the low area DL-PIPO hybrid-triple GNB multiplier holds $E^{(k-1)} = (AB)^{(k-1)}(CD)^{(k-1)} = ABCD$. On the other hand, the high speed version generates its final output after $k+1$ clock cycles, since an extra clock cycle is required, at the beginning, to store the MSD output digit of CD in register $\langle V\rangle$ and the MSD output digit of AB in register $\langle W\rangle$ (see Figure 5b).

C. Space and Time Complexity Analysis

Here, we derive space and time complexities of proposed digit-level hybrid-double and hybrid-triple GNB multipliers. Table IV shows space complexities of proposed hybrid GNB multipliers, while Table V presents their time complexities. In the latter, $T_{PISO} = T_A + (\lceil \log_2 T\rceil + \lceil \log_2 m\rceil)T_X$ denotes the delay in the DL-PISO single GNB multiplier, $T_{SIPO} = T_A + (\lceil \log_2 (d+1)\rceil + \lceil \log_2 T\rceil)T_X$ denotes the delay in the DL-SIPO single GNB multiplier, and $T_{FSIPO} = T_A + (1 + \lceil \log_2 (d+1)\rceil + \lceil \log_2 T\rceil)T_X$ denotes the delay in the DL-FSIPO single GNB multiplier. Notice that, the loading multiplexers listed in Table IV for parallel / serial inputs loading (PIL / SIL) are not shown in Figures 5 and 6.

From Figure 5a, one obtains the space complexity of the low area MSD DL-SIPO hybrid-double GNB multiplier (see Table IV) by adding the space complexities of the MSD DL-PISO and DL-FSIPO single GNB multipliers, in addition to the t AND gates. Similarly, one can find the space complexity of the low area DL-PIPO hybrid-triple GNB multiplier by adding the gate count in its MSD DL-PISO single and DL-SIPO hybrid-double GNB multipliers, in addition to the t AND gates in Figure 6a, as can be seen from Table IV. Moreover, the space complexities of the high speed versions of the proposed hybrid-double and hybrid-triple GNB multipliers are achieved by adding d and $2d$ FFs, respectively, to the space complexities of the corresponding low area versions.

In addition, from Figures 5a and 6a, one finds that the low area architectures of the proposed MSD DL-SIPO hybrid-double GNB multiplier and the proposed DL-PIPO hybrid-triple GNB multiplier offer maximum propagation delays which are equivalent to $T_{PISO}+T_{FSIPO}+T_A$. In the latter formulation, T_{PISO} and T_{FSIPO} denote the propagation delay through the MSD DL-PISO single GNB multiplier and the MSD DL-FSIPO single GNB multiplier, respectively. On the other hand, due to the insertion of registers (Figures 5b and 6b), the propagation delays of the high speed architectures of the proposed digit-level hybrid-double and hybrid-triple GNB multipliers are reduced to $\max\{T_{PISO},T_{FSIPO}+T_A\}$.

The following briefly discusses advantages of digit-level hybrid GNB multipliers for accomplishing double and triple field multiplications, compared to digit-level single GNB multipliers with $2d$ and $3d$ digit sizes, respectively.

D. Hybrid Versus Single Digit-Level GNB Multipliers

The proposed digit-level hybrid-double and hybrid-triple GNB multipliers are constructed using two and three digit-level single GNB multipliers, respectively. Hence, for a fair discussion, we compare the digit-level hybrid-double and hybrid-triple GNB multipliers, of digit size d each, to digit-level single GNB multipliers of digit sizes $2d$ and $3d$, respectively. In this section, we briefly discuss some cases in which using the proposed digit-level hybrid-double and hybrid-triple GNB multipliers is expected to have advantages over using digit-level single GNB multipliers, for accomplishing double

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

and triple field multiplications, respectively. In the future, a more detailed study will be carried out to define the cases in which using the proposed digit-level hybrid multipliers is better, or not, compared to using a digit-level single multiplier.

In some cases, using the digit-level (digit size d) hybrid-double and hybrid-triple GNB multipliers, for double and triple field multiplications, respectively, can achieve lower computational latency or space, and hence higher throughput, compared to using digit-level single GNB multipliers with digit sizes $2d$ and $3d$, respectively.

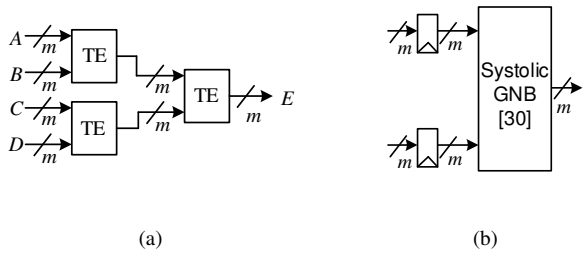
For example, by using a digit-level hybrid-double GNB multiplier with $d = \lceil \frac{m}{3} \rceil$, one obtains the result of a double field multiplication after 3 clock cycles. However, a digit-level single GNB multiplier computes the double field multiplication over 4 clock cycles for a digit size $2d = 2\lceil \frac{m}{3} \rceil$.

As another example, a digit-level hybrid-triple GNB multiplier with $d = \lceil \frac{m}{2} \rceil$ accomplishes a triple field multiplication over 2 clock cycles, while the same latency can only be achieved by using two bit-parallel GNB multipliers which is expected to result in higher space complexity.

E. Hybrid Versus Tree-Structure of Systolic Multiplier in [30]

During review phase of this paper, it has been noted that a tree structure of the systolic single GNB multiplier presented in [30] might lead to triple multiplication with better latency of $4\lceil \sqrt{\frac{m}{d}} \rceil$. The following compares complexities for hybrid-triple multiplication when conducted by our proposed hybrid-triple GNB multiplier versus a tree structure constructed from the systolic multiplier in [30].

The tree structure based on systolic multiplier in [30] is shown in Figure 7a. The tree element (TE) is shown in Figure



(a) (b)

Figure 7: (a) Tree structured triple multiplier based on systolic GNB multiplier in [30]. (b) Tree element (TE).

7b. In Figure 7b, the systolic single GNB multiplier from [30] is constructed out of $\gamma = \left\lceil \sqrt{\lceil \frac{m}{d} \rceil} \right\rceil$ processing elements (PE) of digit size d each. Since the digit size refers to the total number of bits processed in parallel in a single clock cycle [31], [32], therefore, the actual digit size of the multiplier in [30] is γd.

The space complexity of a TE in Figure 7b is obtained by adding two m-bits registers (see Figure 7b) to the multiplier's complexity reported in [30], as follows:

$$FF = 3\left(1 + \left\lceil \sqrt{\frac{m}{d}} \right\rceil\right) m, AND = \left\lceil \sqrt{\frac{m}{d}} \right\rceil dm,$$

$$XOR \leq \frac{\left\lceil \sqrt{\frac{m}{d}} \right\rceil d\,(m-1)}{2}\,(T-1) + \left(1 + \left\lceil \sqrt{\frac{m}{d}} \right\rceil d\right) m.$$

Table VI lists the estimated complexities for hybrid-triple multiplication when $GF\left(2^{163}\right)$ elements are represented by type 4 GNB. Complexities are listed for the tree structured design in Figure 7 and for the proposed low area hybrid-triple GNB multiplier, considering two cases for the latter. In the first case, we increase the digit sizes of our proposed hybrid-triple GNB multiplier to have similar digit size (i.e., comparable area) to the one proposed in [30]. This is similar to what has been commented in [32]. The second case compares the hybrid-triple multiplier to the design of Figure 7 considering comparable latencies.

The comparison in Table VI considers total NAND gate equivalence (GE), throughput (TP), normalized throughput per a NAND gate (TP/G), and Area x Time. In this table, the throughput is estimated as the number of output bits per latency multiplied by a speed of 1 GHz for both listed designs. Time is the latency divided by a speed of 1 GHz (or multiplied by delay of 1 nsec). The total GE is estimated using the STMicroelectronics standard 65nm CMOS library metrics, as explained in Section III-A4 of this paper.

The table shows that the hybrid-triple GNB multiplier achieves lower readings of Area x Time regardless of having comparable area or latency as the design in Figure 7.

Therefore, the proposed DL-PIPO hybrid-triple GNB multiplier accomplishes three field multiplications using the latency required for a single field multiplication of same digit size, at the expense of increasing the area. Hence, it can be used to increase the throughput of applications where such triple multiplications exist. In what follows, we present a new architecture for the eight-ary field exponentiation, as an application for our digit-level hybrid-triple GNB multiplier presented in this section.

VI. Proposed Architecture for Field Exponentiation

Exponentiation is a fundamental operation for the Diffie-Hellman key exchange algorithm [6] and is also used for other cryptographic applications such as random number generation [7]. The n-ary scheme is used to increase throughput of $GF\left(2^m\right)$ exponentiation [24], [23]. In the case of $n = 2^3$ (i.e. eight-ary scheme), to compute the exponentiation A^h for $A \in GF\left(2^m\right)$ and a positive integer $h = \sum_{i=0}^{\lceil \frac{m}{3} \rceil - 1} h_i 2^{3i}$, where $0 \leq h_i < 8$, one rewrites h as $h = \sum_{w=1}^{7} \lambda(w)\, w$, where $\lambda(w) = \sum_{\{i:h_i=w\}} 2^{3i}$. Then, this eight-ary exponentiation scheme requires finding the coefficients $\lambda(w)$ and precomputing and storing odd powers for $1 < w < 8$. This takes at most $\lceil \frac{m}{3} \rceil + 2$ iterations to complete [24], [23]. In this section, we present a new architecture for the eight-ary field exponentiation scheme when the $GF\left(2^m\right)$ elements are represented in the GNB. The proposed architecture is based on the digit-level hybrid-triple GNB multiplier presented in Section V-B (Figure 6) and computes the exponentiation results using $\lceil \frac{m}{3} \rceil$ iterations, while it does not require any storage of precomputed values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

Table VI: Comparing triple multiplication complexities based on proposed hybrid-triple versus design in Figure 7. Throughput is measured at 1 GHz. $\gamma = \left\lceil \sqrt{\left\lceil \frac{m}{d} \right\rceil} \right\rceil d$.

	d	γ	Digit Size	FF	AND	XOR	2 : 1 MUX	Total GE	Latency	TP (Gbps)	TP/G (Kbps/Gate)	Area x Time (Gate x msec)
Low area hybrid-triple (Figure 6a)	9	-	$d = 9$	1123	5803	18675	652	49828.9	19	8.57	172	0.95
	11	-	$d = 11$	1119	7055	22737	652	59440.3	15	10.87	183	0.89
	14	-	$d = 14$	1113	8942	28770	652	73748.2	12	13.58	184	0.88
	45	-	$d = 45$	1051	27349	86895	652	211854.1	4	40.75	192	0.42
	44	-	$d = 44$	1053	26778	85140	652	207666.4	4	40.75	196	0.42
	84	-	$d = 84$	973	47722	149100	652	360419.2	2	81.5	226	0.72
Tree structured systolic GNB multiplier (Figure 7)	9	5	$\gamma d = 45$	8802	22005	55299	-	170011.5	20	8.15	48	3.4
	11	4	$\gamma d = 44$	7335	21516	54081	-	161487.5	16	10.19	63	2.58
	28	3	$\gamma d = 84$	5868	41076	102801	-	276898.2	12	13.58	49	3.32

The following first derives the formulations for field exponentiation, followed by presenting the proposed architecture.

Proposition 3. *Let $F = A^h$ denotes the exponentiation of an arbitrary $GF(2^m)$ element A represented in the GNB, where $1 < h < 2^m$ is an arbitrary positive integer. Therefore, one can compute F using the following recurrence:*

$$F^{(i)} = A^{h_{k'-1-i}} \left(F^{(i-1)}\right)^{2^3}, \tag{8}$$

where $k' = \lceil \frac{m}{3} \rceil$, $h = \sum_{i=0}^{k'-1} h_{k'-1-i} 8^{k'-1-i}$, $0 \le h_{k'-1-i} < 8$, $F^{(-1)} = 1$, and $F = F^{(k'-1)}$.

Proof. By substituting for $i = 0, \ldots, k'-1$ in (8), where $k' = \lceil \frac{m}{3} \rceil$ and $h_{k'-1-i} \in [0, 7]$ are the coefficients of the radix-8 representation of h, we get

$$\begin{aligned} F^{(k'-1)} &= A^{\left(\left(\left(h_{k'-1}8 + h_{k'-2}\right)8 + h_{k'-3}\right)8 + \cdots + h_1\right)8 + h_0} \\ &= A^{\sum_{i=0}^{k'-1} h_{k'-1-i} 8^{k'-1-i}}. \end{aligned}$$

That is, $F^{(k'-1)} = F$, since $h = \sum_{i=0}^{k'-1} h_{k'-1-i} 8^{k'-1-i}$. □

Note that (8) reads h left-to-right. Similarly, one can read h right-to-left by using $F^{(i)} = A^{h_i} \left(F^{(i-1)}\right)^{2^{-3}}$, for $0 \le i < k'$, where $h = 2^{3(k'-1)} \sum_{i=0}^{k'-1} h_i 2^{-3(k'-1-i)}$ and $F = \left(F^{(k'-1)}\right)^{2^{3(k'-1)}}$. Based on (8), we proposed the eight-ary exponentiation architecture of Figure 8, which is constructed based on the proposed digit-level hybrid-triple GNB multiplier of Figure (6) (either Figure 6a or Figure 6b, depending on whether the target application requires a low area design or a high speed design, respectively). As shown in this figure, the architecture is composed of one DL-PIPO hybrid-triple GNB multiplier and four $2 : 1$ m-bits multiplexers. The first three multiplexers (0, 1, and 2), respectively, are controlled by the coefficients $s_0^{(i)}$, $s_1^{(i)}$, and $s_2^{(i)}$ of the binary representation of $h_{k'-1-i} = s_0^{(i)} + s_1^{(i)} 2 + s_2^{(i)} 2^2$, where $k' = \lceil \frac{m}{3} \rceil$ and $0 \le h_{k'-1-i} < 8$ for all $0 \le i < k'$ in (8). The last multiplexer, i.e. 3, passes the field element $1 = (1, \ldots, 1)$ during the first iteration, while it selects the 3-fold right cyclic shift of the multiplier's output during the remaining iterations. Therefore, by using this architecture one computes $F = A^h$ after k' runs of the hybrid-triple multiplication. This is equivalent to $k' \times (L+1)$ clock cycles in the case of parallel preloading of the multiplier, where $L = k$ if a low area hybrid multiplier is used otherwise it becomes $L = k+1$ for using a high speed hybrid multiplier ($k = \lceil \frac{m}{d} \rceil$, d is the digit size).

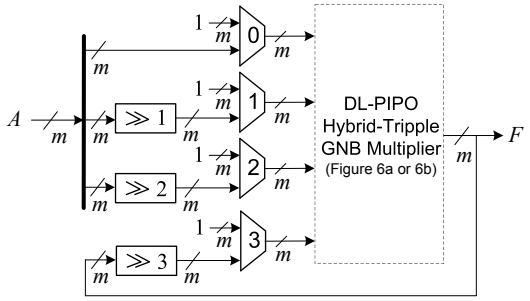


Figure 8: Architecture of the proposed eight-ary field exponentiation scheme. The 1 inputs to multiplexers represent the field element $1 = (1, \ldots, 1)$ represented in the GNB.

Our proposed eight-ary exponentiation architecture does not require any storage of precomputed values, while it has almost the same latency, compared to the existing schemes. Also, it is noted that the proposed architecture uses the same latency regardless of the exponent's value. This in turn prevents leakage of time/power dissipation information.

VII. Conclusion

In this paper, we have proposed three new architectures for digit-level (DL) single multiplication using GNB; two multipliers with fully serial-in-parallel-out (FSIPO) and one with parallel-in-serial-out (PISO). The two DL-FSIPO single GNB multipliers have been proposed for the first time in the literature. They do not require preloading of inputs and, hence, are advantageous for applications where parallel loading of inputs is not possible due to limited size of data-path.

Using the proposed single digit-level multiplier architectures, we have proposed a new digit-level serial-in-parallel-out (DL-SIPO) hybrid-double GNB multiplier and for the first time in the literature a new digit-level parallel-in-parallel-out (DL-PIPO) hybrid-triple GNB multiplier. The proposed digit-level hybrid-double and hybrid-triple multipliers, perform two and three field multiplications, respectively, using the same latency as a single digit-level field multiplication, at the expense of more area.

As an application of the proposed hybrid-triple multiplier, we have presented a new digit-level eight-ary field exponentiation architecture which offers computational latency similar to the existing eight-ary schemes, however, without requiring storage of precomputed values.

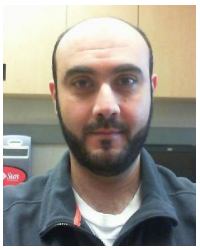
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2481408, IEEE Transactions on Computers

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their valuable constructive comments.

REFERENCES

[1] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications*. New York, NY, USA: Cambridge University Press, 1986.
[2] J. Deschamps, J. Imaña, and G. Sutter, *Hardware Implementation of Finite-Field Arithmetic*, ser. Electronic engineering. McGraw-Hill Education, 2009.
[3] H. El-Razouk, A. Reyhani-Masoleh, and G. Gong, "New Implementations of the WG Stream Cipher," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to appear.
[4] ——, "New Hardware Implementations of WG(29,11) and WG-16 Stream Ciphers Using Polynomial Basis," *IEEE Trans. Comput.*, to appear.
[5] "IEEE Standard Specifications for Public-Key Cryptography," *IEEE Std 1363-2000*, p. i, 2000.
[6] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.
[7] C. Wang and D. Pei, "A VLSI Design for Computing Exponentiations in $GF(2^m)$ and its Application to Generate Pseudorandom Number Sequences," *IEEE Transactions on Computers*, vol. 39, no. 2, pp. 258–262, Feb. 1990.
[8] A. Reyhani-Masoleh and M. Hasan, "A New Construction of Massey-Omura Parallel Multiplier Over $GF(2^m)$," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 511 –520, May. 2002.
[9] *Digital Signature Standard (DSS)*, Federal Information Processing Standards (FIPS), U.S. National Institute of Standards and Technology (NIST) Std. FIPS 186-4, July 2013.
[10] J. L. Massey and J. K. Omura, "Computational Method and Apparatus for Finite Field Arithmetic," US Patent 4 587 627, May 6, 1986.
[11] C. Wang, T. Troung, H. Shao, L. Deutsch, J. Omura, and I. S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)," *IEEE Trans. Comput.*, vol. C-34, no. 8, pp. 709–717, 1985.
[12] M. Hasan, M. Wang, and V. Bhargava, "A Modified Massey-Omura Parallel Multiplier for a Class of Finite Fields," *IEEE Trans. Comput.*, vol. 42, no. 10, pp. 1278 –1280, Oct. 1993.
[13] C. Koc and B. Sunar, "Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields," *IEEE Trans. Comput.*, vol. 47, no. 3, pp. 353–356, 1998.
[14] A. Reyhani-Masoleh and M. Hasan, "Low Complexity Word-Level Sequential Normal Basis Multipliers," *IEEE Trans. Comput.*, vol. 54, no. 2, pp. 98–110, 2005.
[15] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone, "An Implementation for a Fast Public-Key Cryptosystem," *J. Cryptology*, vol. 3, pp. 63–79, 1991.
[16] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989.
[17] L. Gao and G. Sobelman, "Improved VLSI Designs for Multiplication and Inversion in GF(2^m) Over Normal Bases," in *ASIC/SOC Conference, 2000. Proceedings. 13th Annual IEEE International*, 2000, pp. 97–101.
[18] W. Geiselmann and D. Gollmann, "Symmetry and Duality in Normal Basis Multiplication," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, ser. Lecture Notes in Computer Science, T. Mora, Ed. Springer Berlin Heidelberg, 1989, vol. 357, pp. 230–238.
[19] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *Selected Areas in Communications, IEEE Journal on*, vol. 7, no. 4, pp. 458–466, 1989.
[20] A. Reyhani-Masoleh and M. A. Hasan, "Efficient Digit-Serial Normal Basis Multipliers Over Binary Extension Fields," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 575–592, Aug. 2004.
[21] A. Reyhani-Masoleh, "Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases," *IEEE Trans. Comput.*, vol. 55, no. 1, pp. 34–47, Jan 2006.
[22] R. Azarderakhsh and A. Reyhani-Masoleh, "Low-Complexity Multiplier Architectures for Single and Hybrid-Double Multiplications in Gaussian Normal Bases," *IEEE Trans. Comput.*, vol. 62, no. 4, pp. 744–757, April 2013.
[23] D. Stinson, "Some Observations on Parallel Algorithms for Fast Exponentiation in $GF(2^n)$," *SIAM J. Comput.*, vol. 19, no. 4, pp. 711–717, Aug. 1990.
[24] D. M. Gordon, "A Survey of Fast Exponentiation Methods," *Journal of Algorithms*, vol. 27, no. 1, pp. 129–146, Apr. 1998.
[25] R. Azarderakhsh and A. Reyhani-Masoleh, "A Modified Low Complexity Digit-Level Gaussian Normal Basis Multiplier," in *Arithmetic of Finite Fields*, ser. Lecture Notes in Computer Science, M. Hasan and T. Helleseth, Eds. Springer Berlin Heidelberg, 2010, vol. 6087, pp. 25–40.
[26] C. Paar, "Optimized Arithmetic for Reed-Solomon Encoders," in , *1997 IEEE International Symposium on Information Theory. 1997. Proceedings*, Jun. 1997, pp. 250–.
[27] Y. Chen and K. K. Parhi, "Small Area Parallel Chien Search Architectures for Long BCH Codes," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 5, pp. 545–549, May 2004.
[28] The Sage Notebook, http://www.sagenb.org/.
[29] J.-S. Horng, I.-C. Jou, and C.-Y. Lee, "Low-Complexity Multiplexer-Based Normal Basis Multiplier Over GF(2^m)," *Journal of Zhejiang University SCIENCE A*, vol. 10, no. 6, pp. 834–842, May 2009.
[30] R. Azarderakhsh, M. Kermani, S. Bayat-Sarmadi, and C.-Y. Lee, "Systolic Gaussian Normal Basis Multiplier Architectures Suitable for High-Performance Applications," *IEEE Trans. Very Large Scale Integr. Syst.*, 2014 (to appear).
[31] K. Parhi, "A Systematic Approach for Design of Digit-Serial Signal Processing Architectures," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 4, pp. 358–375, Apr. 1991.
[32] A. Reyhani-Masoleh, "Comments on "Low-Latency Digit-Serial Systolic Double Basis Multiplier over GF(2^m) Using Subquadratic Toeplitz Matrix-Vector Product Approach"," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 1215–1216, April 2015.



Hayssam El-Razouk received the BEng degree from Beirut Arab University in 2002 with the first rank, and the MESc degree from the University of Western Ontario in 2006, both in electrical and computer engineering. From 2006 to 2011, he was a software engineer with RedIron Technologies, London, Ontario, Canada. He joined the University of Western Ontario, Canada, in 2011 as a PhD student. Recently in summer 2015, he completed his PhD degree in electrical and computer engineering. During PhD studies, he was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada (NSERC) Alexander Graham Bell Canada Graduate Scholarships-Doctoral (CGS-D scholarship, September 2012 to August 2015). Currently, he is a research associate with the electrical and computer engineering department at the University of Western Ontario, Canada. His research interests include digital VLSI circuits design, defect tolerance in VLSI digital circuits, finite fields and cryptographic hardware, and pseudo random sequences.

Arash Reyhani-Masoleh received the BSc degree in electrical and electronic engineering from Iran University of Science and Technology in 1989, the MSc degree in electrical and electronic engineering from the University of Tehran in 1991, both with the first rank, and the PhD degree in electrical and computer engineering from the University of Waterloo in 2001. From 1991 to 1997, he was with the Department of Electrical Engineering, Iran University of Science and Technology. From June 2001 to September 2004, he was with the Center for Applied Cryptographic Research, University of Waterloo, where he was awarded a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship in 2002. In October 2004, he joined the Department of Electrical and Computer Engineering, Western University, London, Canada, where he is currently a tenured associate professor. His current research interests include fault-tolerant computing, algorithms and VLSI architectures for computations in finite fields, cryptography, and error control coding. He has been a two-time recipient of NSERC Discovery Accelerator Supplement (DAS) award in 2010 and 2015. Currently, he serves as an associate editor for Integration, the VLSI Journal (Elsevier). He is a member of the IEEE and the IEEE Computer Society.