We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

Scalable and Systolic Gaussian Normal Basis Multipliers over GF(2^m) Using Hankel Matrix-Vector Representation

Chiou-Yng Lee
Lunghwa University of Science and Technology
Taoyuan County, Taiwan

1. Introduction

Efficient design and implementation of finite field multipliers have received high attention in recent years because of their applications in elliptic curve cryptography (ECC) and error control coding (Denning, 1983; Rhee, 1994; Menezes, Oorschot & Vanstone, 1997). Although channel codes and cryptographic algorithms both make use of the finite field GF(2^m), the field orders needed differ dramatically: channel codes are typically restricted to arithmetic with field elements which are represented by up to eight bits, whereas ECC rely on field sizes of several hundred bits. The majority of publications concentrate on finite field architectures for relatively small fields suitable for implementation of channel codes. In finite field GF(2^m), multiplication is one of the most important and time-consuming computations. Since cryptographic applications (Menezes, Oorschot & Vanstone, 1997) are the Diffie-Hellman key exchange algorithm based on the discrete exponentiation over GF(2^m), the methods of computing exponentiation over GF(2^m) based on Fermat's theorem are performed by the repeated multiply-square algorithm. Therefore, to provide the high performance of the security function, the efficient design of high-speed algorithms and hardware architectures for computing multiplication is required and considered.

There are three popular basis representations, termed polynomial basis (PB), normal basis (NB), and dual basis (DB). Each basis representation has its own advantages. The normal basis multiplication is generally selected for cryptography applications, because the squaring of the element in GF(2^m) is simply the right cyclic shift of its coordinates. NB multiplication depended the selection of key function is discovered by Massey and Omura (1986). For the elliptic curve digital signature algorithm (ECDSA) in IEEE Standard P1363 (2000) and National Institute of Standards and Technology (NIST) (2000), Gaussian normal basis (GNB) is defined to implement the field arithmetic operation. The GNB is a special class of normal basis, which exists for every positive integer m not divisible by eight. The GNB for GF(2^m) is determined by an integer t, and is called the type-t Gaussian normal basis. However, the complexity of a type-t GNB multiplier is proportional to t (Reyhani-Masoleh, 2006), small values of t are generally chosen to ensure that the field multiplication is implemented efficiently.

Among various finite field multipliers are classified either as a parallel or serial architectures. Bit-serial multipliers (Reyhani-Masoleh & Hasan, 2005; Lee & Chang, 2004) require less area, but are slow that is taken by m clock cycles to carry out the multiplication of two elements. Conversely, bit-parallel multipliers (Lee, Lu & Lee, 2001; Hasan, Wang & Bhargava, 1993; Kwon, 2003; Lee & Chiou, 2005) tend to be faster, but have higher hardware costs. Recently, various multipliers (Lee[1], 2003; Lee, Horng & Jou, 2005; Lee, 2005; Lee[2], 2003) focus on bit-parallel architectures with optimal gate count. However, previously mentioned bit-parallel multipliers show a computational complexity of $O(m^2)$ operations in GF(2); canonical and dual basis architectures are lower bounded by m^2 multiplications and $k^2 - 1$ additions, normal basis ones by k^2 multiplications and $2k^2 - k$ additions. A multiplication in GF(2) can be realized by a two-input AND gate and an adder by a two-input XOR gate. For this reason, it is attractive to provide architectures with low computational complexity for efficient hardware implementations.

There have digit-serial/scalable multiplier architectures to enhance the trade-off between throughput performance and hardware complexity. The scalable architecture needs both the element A and B are separated into $n=[m/d]$ sub-word data, while the digit-serial architecture only requires one of the element to separate sub-word data. Both architectures are used by the feature of the scalability to handle growing amounts of work in a graceful manner, or to be readily enlarged. In (Tenca & Koc, 1999), a unit is considered scalable defined that the unit can be reused or replicated in order to generate long-precision results independently of the data path precision for which the unit was originally designed. Various digit-serial multipliers are recently developed in (Paar, Fleischmann & Soria-Rodriguez, 1999; Kim & Yoo, 2005; Kim, Hong and Kwon, 2005; Guo & Wang, 1998; Song & Parhi, 1998; Reyhani-Masoleh & Hasan, 2002). Song and Parhi (1998) proposed MSD-first and LSD-first digit-serial PB multipliers using Horner's rule scheme. For partitioning the structure of two-dimension arrays, efficient digit-serial PB multipliers are found in (Kim & Yoo, 2005; Kim, Hong & Kwon, 2005; Guo & Wang, 1998). The major feature of these architectures is combined with both serial and parallel algorithms.

For large word lengths commonly found in cryptography, the bit-serial approach is rather slow, while bit-parallel realization requires large circuit area and power consumption. In elliptic curve cryptosystems, it strongly depends on the implementation of finite field arithmetic. By employing the Hankel matrix-vector representation, the new GNB multiplication algorithm over GF(2^m) is presented. Utilizing the basic characteristics of MSD-first and LSD-first schemes (Song & Parhi, 1998), it is shown that the proposed GNB multiplication can be decomposed into $n(n+1)$ Hankel matrix-vector multiplications. The proposed scalable GNB multipliers are including one $d \times d$ Hankel multiplier, two registers and one final reduction polynomial circuit. The results reveal that, if the selected digital size is $d \geq 4$ bits, the proposed architecture has less time-space complexity than traditional digit-serial systolic multipliers, and can thus obtain an optimum architecture for GNB multiplier over GF(2^m). To further saving both time and space complexities, the proposed scalar multiplication algorithm with Hankel matrix-vector representation can also be realized by a scalable and systolic architecture for polynomial basis and dual basis of GF(2^m).

The rest of this paper is structured as follows. Section 2 briefly reviews a conventional NB multiplication algorithm and a Hankel matrix-vector multiplication. Section 3 proposes the two GNB multipliers, based on Hankel matrix-vector representation, to yield a scalable and systolic architecture. Section 4 introduces the modified GNB multiplier. Section 5 analyzes

our proposed GNB multiplier in the term of the time-area complexity. Finally, conclusions are drawn in Section 6.

2. Preliminaries

2.1 Gaussian normal basis multiplication

The finite field GF(2^m) is well known to be viewable as a vector space of dimension m over GF(2). A set $N = \{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ is called the normal basis of GF(2^m), and α is called the normal element of GF(2^m). Let any element $A \in$ GF(2^m) can be represented as

$$A = \sum_{i=0}^{m-1} a_i \alpha^{2^i} = (a_0, a_1, \cdots, a_{m-1}) \tag{1}$$

where $a_i \in$ GF(2), $0 \le i \le m-1$, denotes the ith coordinate of A. In hardware implementation, the squaring element is performed by a cyclic shift of its binary representation. The multiplication of elements in GF(2^m) is uniquely determined by the m cross products $\alpha\alpha^{2^i} = \sum_{j=0}^{m-1} \mu_{ij}\alpha^{2^j}$, $\mu_{ij} \in$ GF(2). $\mathbf{M}=\{\mu_{ij}\}$ is called a multiplication matrix. Let $A=(a_0, a_1, \ldots, a_{m-1})$ and $B=(b_0, b_1, \ldots, b_{m-1})$ indicate two normal basis elements in GF(2^m), and $C=(c_0, c_1, \ldots, c_{m-1}) \in$ GF(2^m) represent their product, i.e., $C=AB$. Coordinate ci of C can then be represented by

$$c_i = A^{(i)}\mathbf{M}(B^{(i)})^T \tag{2}$$

where $A^{(i)}$ denotes a right cyclic shift of the element A by i positions. To compute the multiplication matrix **M**, one can see in (IEEE Standard P1363, 2000; Reyhani-Masoleh & Hasan, 2003). When the multiplication matrix M is found, the NB multiplication algorithm is described as follows:

Algorithm 1: (NB multiplication) (IEEE Standard P1363, 2000)
Input: $A=(a_0, a_1, \ldots, a_{m-1})$ and $B=(b_0, b_1, \ldots, b_{m-1}) \in$ GF(2^m)
Output: $C=(c_0, c_1, \ldots, c_{m-1})=AB$
1. initial: $C=0$
2. for $i = 0$ to $m - 1$ {
3. $c_i = A\mathbf{M}B^T$
4. $A=A^{(1)}$ and $B=B^{(1)}$
5. }
6. output $C=(c_0, c_1, \ldots, c_{m-1})$

Applying Algorithm 1, Massey and Omura (1986) first proposed bit-serial NB multiplier. The complexity of the normal basis N, represented by C_N, is the number of nonzero μ_{ij} values in **M**, and determines the gate count and time delay of the NB multiplier. It is shown in (Mullin, Onyszchuk, Vanstone & Wilson, 1988/1989) that C_N for any normal basis of GF(2^m) is greater or equal to $2m-1$. In order to an efficient and simple implementation, a normal basis is chosen that C_N is as small as possible. Two types of an optimal normal basis (ONB), type-1 and type-2, exist in GF(2^m) if $C_N = 2m-1$. However, such ONBs do not exist for all m.

Definition 1. Let $p=mt+1$ represent a prime number and $gcd(mt/k,m)=1$, where k denotes the multiplicative order of 2 module p. Let γ be a primitive p root of unity. The type-t Gaussian

normal basis (GNB) is employed by $\alpha=\gamma+\gamma^{2^m}+\gamma^{2^{2m}}+...+\gamma^{2^{(t-1)m}}$ to generate a normal basis N for GF(2^m) over GF(2).

Significantly, GNBs exist for GF(2^m) whenever m is not divisible by 8. By adopting Definition 1, each element A of GF(2^m) can also be given as

$$\begin{aligned} A &= a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-1}} = a_0(\gamma + \gamma^{2^m} + \cdots + \gamma^{2^{m(t-1)}}) \\ &+ a_1(\gamma^2 + \gamma^{2^{m+1}} + \cdots + \gamma^{2^{m(t-1)+1}}) + \cdots + a_{m-1}(\gamma^{2^{m-1}} + \gamma^{2^{2m-1}} + \cdots + \gamma^{2^{mt-1}}) \end{aligned} \tag{3}$$

From Equation (3), the type-t GNB can be represented by the set $\{\gamma,\gamma^2,\cdots,\gamma^{2^{m-1}},\gamma^{2^m},\gamma^{2^{m+1}},\cdots,\gamma^{2^{2m-1}},\cdots,\gamma^{2^{m(t-1)}},\gamma^{2^{m(t-1)+1}},\cdots,\gamma^{2^{mt-1}}\}$. Since γ is a primitive p root of unity, we have

$$\gamma\gamma^i = \begin{cases} \gamma^{i+1} & if \quad i \neq p-1 \\ 1 & if \quad i = p-1 \end{cases}. \tag{4}$$

Thus, the GNB can then alternate to represent the redundant basis $R=\{\gamma, \gamma^2,\ldots, \gamma^{p-1}\}$. The field element A can be alternated to define the following formula:

$$A = a_{F(1)}\gamma + a_{F(2)}\gamma^2 + \cdots + a_{F(p-1)}\gamma^{p-1} \tag{5}$$

where

$F(2^i 2^{mj} \bmod p)=i$ for $0 \le i \le m-1$ and $0 \le j \le t-1$.

Example 1. For example, Let $A=(a_0,a_1,a_2,a_3,a_4)$ be the NB element of GF(2^5), and let $\alpha=\gamma+\gamma^{10}$ be used to generate the NB. Applying the redundant representation, the field element A can be represented by $A=a_0\gamma+a_1\gamma^2+a_3\gamma^3+a_2\gamma^4+a_4\gamma^5+a_4\gamma^6+a_2\gamma^7+a_3\gamma^8+a_1\gamma^9+a_0\gamma^{10}$.

Observing this representation, the coefficients of A are duplicated by t-term coefficients of the original normal basis element $A=(a_0, a_1,\ldots, a_{m-1})$ if the field element A presents a type-t normal basis of GF(2^m). Thus, by using the function $F(2^i 2^{mj} \bmod p)=i$, the field element $A=(a_{F(1)},a_{F(2)},\cdots,a_{F(p-1)})$ with the redundant representation can be translated into the following representation, $A=(\underbrace{a_0,\cdots,a_0}_{t},a_1,\cdots a_1,a_2,\cdots,a_2,\cdots\cdots a_{m-1},\cdots,a_{m-1})$. Therefore, the redundant basis is easy converted into the normal basis element, and is without extra hardware implementations.

Let $A=(a_0, a_1,\ldots, a_{m-1})$ and $B=(b_0, b_1,\ldots, b_{m-1})$ indicate two normal basis elements in GF(2^m), and $C=(c_0, c_1,\ldots, c_{m-1})$ represent their product, i.e., $C=AB$. Coordinate ci of C can then be calculated as in the following formula: (IEEE Standard P1363, 2000)

$$c_0 = G(A,B) = \sum_{j=1}^{p-2} a_{F(j+1)} b_{F(p-j)} \tag{6}$$

According to Algorithm 1, the GNB multiplication algorithm for even t is modified as follows.

Algorithm 2: (GNB multiplication for t even) (IEEE Standard P1363, 2000)

Input: $A=(a_0,a_1,\ldots,a_{m-1})$ and $B=(b_0,b_1,\ldots,b_{m-1})\in$GF($2^m$)

Output: $C=(c_0,c_1,\ldots,c_{m-1})=AB$

1. initial : $C=0$
2. for $k=0$ to $m-1$ {
3. $c_k= G(A,B)$
4. $A=A^{(1)}$ and $B=B^{(1)}$

5. }
6. Output $C=(c_0,c_1,\ldots,c_{m-1})$

2.2 Bit-parallel systolic Hankel multiplier

This section briefly introduces a bit-parallel systolic Hankel multiplier in (Lee & Chiou, 2005).

Definition 2. An $m\times m$ matrix H is known as the Hankel matrix if it satisfies the relation $H(i,j)=H(i+1,j-1)$, for $0 \le i \le m-2$ and $1 \le j < m-1$, where $H(i,j)$ represents the element in the intersection of row i and column j, such that

$$\mathbf{H}=\begin{bmatrix} h_0 & h_1 & \cdots & h_{m-3} & h_{m-2} & h_{m-1} \\ h_1 & h_2 & \iddots & \iddots & h_{m-1} & h_m \\ h_2 & h_3 & \iddots & \iddots & \iddots & h_{m+1} \\ \vdots & \vdots & \iddots & \iddots & \iddots & \vdots \\ h_{m-2} & h_{m-1} & \iddots & \iddots & h_{2m-4} & h_{2m-3} \\ h_{m-1} & h_m & \cdots & h_{2m-4} & h_{2m-3} & h_{2m-2} \end{bmatrix}.$$

A Hankel matrix **H** is entirely determined by its last column and first row, and thus depends on having $2m-1$ parameters, i.e., $h=(h_0, h_1,\ldots, h_{m-2}, h_{m-1}, h_m,\ldots, h_{2m-3}, h_{2m-2})$. The entries of **H** are constant down the diagonals parallel to the main anti-diagonal. Let $C=hB$ be the product of h and B, where $B=(b_0,b_1,\ldots,b_{m-1})$. The coordinate c_i of C is given by

$$c_i=\sum_{j=0}^{m-1} h_{j+i}b_j \tag{7}$$

Definition 3. Let i,j,m be these positive integers with $0 \le i,j \le m-1$, one can define the following function $\sigma(i,j)$:

$$\sigma(i,j)=\begin{cases} <\dfrac{j-i}{2}> & for \quad i+j=even \\ <-\dfrac{j+i+1}{2}> & for \quad i+j=odd \end{cases}$$

where $<q>$ denotes q mod m.

Let i denote a fixed integer in the complete set $\{1,2,\ldots,m-1\}$, one verifies that the map $k=\sigma(i,j)$. For instance, Table 1 presents the relationship between j and k with $k=\sigma(i,j)$, where $m=7$, $i=2$, and $0 \le j \le 6$. Therefore, by substituting $j=\sigma(i,j)$ into Equation (6), the product C can be denoted as

$$C=\sum_{i=0}^{m-1}\left(\sum_{j=0}^{m-1} h_{\sigma(i,j)+i}b_{\sigma(i,j)}\right)\gamma^i \tag{8}$$

j	0	1	2	3	4	5	6
k=σ(i,j)	6	5	0	4	1	3	2

Table 1. The relationship between j and k for i=2

Example 2: Let $A=(a_0, a_1, a_2, a_3, a_4)$ and $h=(h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8)$ represent two vectors and **H** represents the Hankel matrix defined by h; and let $C=(c_0, c_1, c_2, c_3, c_4)$ be the product of hA. By applying Equation (8), the product can be derived as follows.

$$\begin{array}{rccccc} & 1 & x & x^2 & x^3 & x^4 \\ & a_0h_0 & a_4h_5 & a_4h_6 & a_3h_6 & a_3h_7 \\ & a_4h_4 & a_0h_1 & a_3h_5 & a_4h_7 & a_2h_6 \\ & a_1h_1 & a_3h_4 & a_0h_2 & a_2h_5 & a_4h_8 \\ & a_3h_3 & a_1h_2 & a_2h_4 & a_0h_3 & a_1h_5 \\ + & a_2h_2 & a_2h_3 & a_1h_3 & a_1h_4 & a_0h_4 \\ \hline & c_0 & c_1 & c_2 & c_3 & c_4 \end{array}$$

Figure 1 depicts the bit-parallel systolic Hankel multiplier given by Example 1. The multiplier comprises 25 cells, including 14 U-cells and 11 V-cells. Every $U_{i,j}$ cell (Figure 2(a)) contains one 2-input AND gate, one 2-input XOR gate and three 1-bit latches to realize $c_i=c_i+a_{\sigma(i,j)}h_{\sigma(i,j)+i}$. Each $V_{i,j}$ cell (Figure 2(b)) is formed by one 2-input AND gate, one 2-input XOR gate and four 1-bit latches to realize the operation of $c_i=c_i+a_{\sigma(i,j)}h_{\sigma(i,j)+i}$, or $c_i=c_i+a_{\sigma(i,j)}h_{\sigma(i,j)+i+m}$. As stated in the above cell operations, the latency needs m clock cycles, and the computation time per cell is needed by one 2-input AND gate and one 2-input XOR gate.

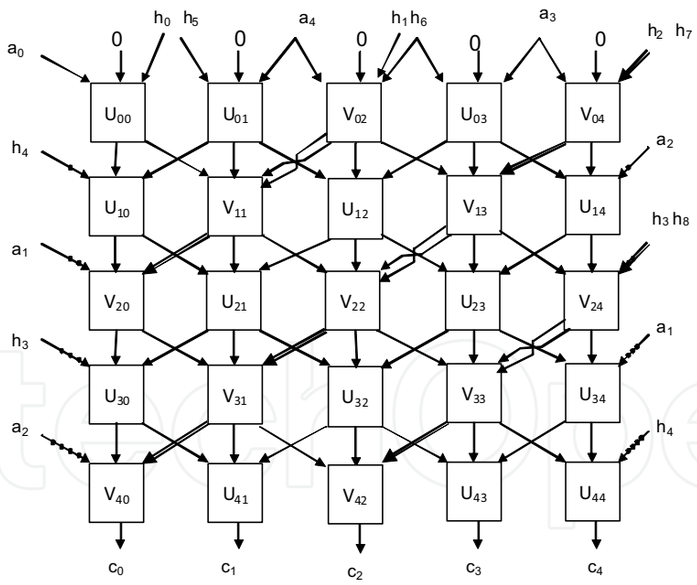

Fig. 1. The bit-parallel systolic Hankel multiplier [10]

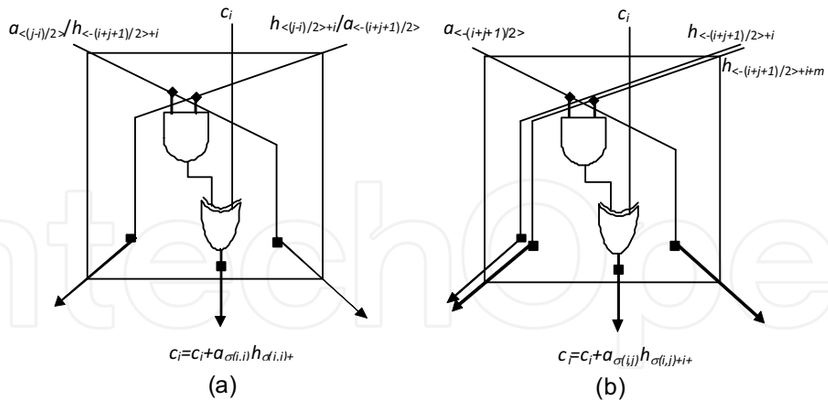

Fig. 2. (a) The detailed circuit of U-cell; (b) The detailed circuit of V-cell

3. Proposed scalable and systolic GNB multiplier architectures

Let $A = \sum_{i=0}^{p-1} a_{F(i)}\gamma^i$ and $B = \sum_{i=0}^{p-1} b_{F(i)}\gamma^i$ with $a_{F(0)}=b_{F(0)}=0$ and $a_{F(i)}$, $b_{F(i)}\in$GF(2) for $1\le i \le p-1$ denote two type-t GNB elements in GF(2^m), where γ represents the root of x^p+1. Assume that the chosen digital size is a d-bit, and $n=\lceil p/d \rceil$, both elements A and B can also be expressed as follows.

$$A = \sum_{i=0}^{n-1} A_i\gamma^{di} \ , \ B = \sum_{i=0}^{n-1} B_i\gamma^{di}$$

where

$$A_i = \sum_{j=0}^{d-1} a_{F(di+j)}\gamma^j \ , B_i = \sum_{j=0}^{d-1} b_{F(di+j)}\gamma^j \ .$$

Based on the partial multiplication for determining AB_0, the partial product can be denoted by

$$AB_0= A_0B_0 + A_1B_0\gamma^d+\ldots+ A_{n-1}B_0\gamma^{d(n-1)} \quad (9)$$

Each term A_iB_0 of degree $2d-2$ is the core computation of Equation (9). In a general multiplication, let us define that AiB0 is formed by

$$A_iB_0=S_i+D_i\gamma^d, \text{ for } 0 \le i \le n-1. \quad (10)$$

where

$$S_i=s_{i,0}+s_{i,1}\gamma+\ldots+s_{i,d-1}\gamma^{d-1}$$
$$D_i=d_{i,0}+d_{i,1\gamma}+\ldots+d_{i,d-1}\gamma^{d-2}$$
$$s_{i,j}= \sum_{k=0}^{j} a_{F(id+k)}b_{F(j-k)} \text{ , for } 0 \le j \le d-1$$
$$d_{i,j}= \sum_{k=j+1}^{d-1} a_{F(id+k)}b_{F(d+j-k)} \text{ , for } 0 \le j \le d-2$$

Therefore, Equation (9) can be re-expressed as
$AB_0=(S_0+D_0\gamma^d)+ (S_1+D_1\gamma^d)\ \gamma^d +\ldots+ (S_{n-1}+D_{n-1}\gamma^d)\ \gamma^{d(n-1)}$

$= S_0+(S_1+D_0)\ \gamma^d +...+ (S_{n-1}+D_{n-2})\ \gamma^{d(n-1)}+ D_{n-1}\gamma^{dn}$

$$= C_0+C_1\gamma^d +...+ C_{n-1}\gamma^{d(n-1)} + C_n\gamma^{dn} \tag{11}$$

where

$C_0= S_0$,

$C_i = S_i+D_{i-1}$, for $1 \le i \le n-1$

$C_n= D_{n-1}$

$S_i=(s_{i,0}, s_{i,1} ,..., s_{i,d-1})$ in Equation (10) can be translated with the following matrix-vector

$$\begin{bmatrix} s_{i,0} \\ s_{i,1} \\ \vdots \\ s_{i,d-2} \\ s_{i,d-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & a_{F(id)} \\ 0 & 0 & \iddots & a_{F(id)} & a_{F(id+1)} \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ 0 & a_{F(id)} & \cdots & a_{F(id+d-3)} & a_{F(id+d-2)} \\ a_{F(id)} & a_{F(id+1)} & \cdots & a_{F(id+d-2)} & a_{F(id+d-1)} \end{bmatrix} \begin{bmatrix} b_{F(d-1)} \\ b_{F(d-2)} \\ \vdots \\ b_{F(1)} \\ b_{F(0)} \end{bmatrix} \tag{12}$$

Similarly, $D_{i-1}=(d_{i-1,0}, d_{i-1,1} ,..., d_{i-1,d-2}, 0)$ can also be translated with the following matrix-vector

$$\begin{bmatrix} d_{i-1,0} \\ d_{i-1,1} \\ \vdots \\ d_{i-1,d-2} \\ 0 \end{bmatrix} = \begin{bmatrix} a_{F(d(i-1)+1)} & \cdots & a_{F(id-2)} & a_{F(id-1)} & 0 \\ a_{F(d(i-1)+2)} & \cdots & a_{F(id-1)} & 0 & 0 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ a_{F(id-1)} & \iddots & \iddots & \iddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} b_{F(d-1)} \\ b_{F(d-2)} \\ \vdots \\ b_{F(1)} \\ b_{F(0)} \end{bmatrix} \tag{13}$$

According to Equations (12) and (13), $C_i=S_i+D_{i-1}$ in Equation (11) can obtain

$$\begin{bmatrix} c_{F(di)} \\ c_{F(di+1)} \\ \vdots \\ c_{F(d(i+1)-1)} \end{bmatrix} = \begin{bmatrix} a_{F(d(i-1)+1)} & a_{F(d(i-1)+2)} & \cdots & a_{F(id)} \\ a_{F(d(i-1)+2)} & a_{F(d(i-1)+3)} & \cdots & a_{F(id+1)} \\ \vdots & \vdots & \iddots & \vdots \\ a_{F(di)} & a_{F(di+1)} & \cdots & a_{F(id+d-1)} \end{bmatrix} \begin{bmatrix} b_{F(d-1)} \\ b_{F(d-2)} \\ \vdots \\ b_{F(0)} \end{bmatrix} \tag{14}$$

$$= \mathbf{H}_{n-i}B_0$$

From the above matrix-vector multiplication, the Hankel vector $h_{n-i}=(a_{F(d(i-1)+1)}, a_{F(d(i-1)+2)},\ldots, a_{F(id)}, a_{F(id+1)},\ldots, a_{F(d(i+1)-1)})$ is defined by the $d\times d$ Hankel matrix H_{n-i}. Hence, the computation AB_0 using the Hankel matrix-vector representation can be computed as follows:

$$AB_0=h_nB_0+ h_{n-1}B_0\gamma^d+\ldots+h_0B_0\gamma^{dn} \tag{15}$$

Therefore, AB_0 can be dismembered into $(n+1)$ Hankel multiplications, and can be performed by the following algorithm:

Algorithm 3: $PM(A,B_0)$

Input: $A=\sum_{i=0}^{p-1} a_{F(i)}\gamma^j$ and $B_0 = \sum_{i=0}^{d-1} b_{F(i)}\gamma^i$

Output: $C = \sum_{i=0}^{p-1} c_{F(i)}\gamma^i = AB_0 \bmod (\gamma^p+1)$

1. convert the Hankel vector $h_{n-i}=(a_{F(d(i-1)+1)}, a_{F(d(i-1)+2)},\ldots, a_{F(d(i+1)-1)})$, for $0\le i \le n$, from A.
2. $C=(c_{F(0)}, c_{F(1)},\ldots, c_{F(p-1)})=(0,\ldots,0)$
3. for i=0 to n {
4. $X_i=h_{n-i}B_0$
5. }

6. $C = X \bmod (\gamma^p+1)$
7. return $C=(c_{F(0)}, c_{F(1)},\ldots, c_{F(p-1)})$

Algorithm 3 for determining AB_0 includes two core operations, namely the Hankel multiplication and the reduction polynomial γ^p+1, as illustrated in Figure 3. The proposed partial multiplier architecture in Figure 3 can be calculated using the following procedure.

Step 1: From Equation (14), we can see that Hankel matrix H_{n-i} is defined by all coefficients in A. Here $A=(a_{F(0)}, a_{F(1)},\ldots, a_{F(p-1)})$ is firstly converted to the Hankel vector $h_{n-i}=(a_{F(d(i-1)+1)}, a_{F(d(i-1)+2)},\ldots, a_{F(d(i+1)-1)})$ and its result is stored in the register H_{n-i}.

Step 2: Applying the bit-parallel systolic Hankel multiplier as shown in Figure1, Figure 3 shows AB_0 computation. Each Hankel multiplications in Step 4 of Algorithm 3, the result of AB_0 is stored in the register X.

Step 3: After (n+1) Hankel multiplications, the result needs to perform the reduction polynomial γ^p+1.

Generally, the computation of AB_i for $0 \le i \le n-1$ can be obtained by the following formula:

$$AB_i = h_n B_i + h_{n-1} B_i \gamma^d + \ldots + h_0 B_i \gamma^{dn} \quad (16)$$

The above equation indicates that each AB_i computation can be dismembered into (n+1) Hankel multiplications. As mentioned above, two GNB multipliers are described in the following subsections.

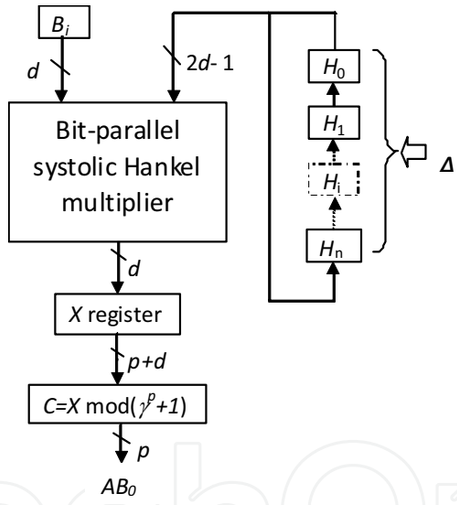

Fig. 3. The proposed scalable and systolic architecture for computing AB0

3.1 LSD-first scalable systolic multiplier

The product $C = AB \bmod (\gamma^p+1)$ using LSD-first multiplication algorithm can be represented as

$$C = AB_0 \bmod (\gamma^p+1) + AB_1\gamma^d \bmod (\gamma^p+1) + \ldots + AB_{n-1}\gamma^{d(n-1)} \bmod (\gamma^p+1)$$
$$= A^{(0)}B_0 + A^{(d)}B_1 + \ldots + A^{(d(n-1))}B_{n-1} \quad (17)$$

where $A^{(id)} = A\gamma^{id} \bmod(\gamma^p+1) = A^{(d(i-1))}\gamma^d \bmod(\gamma^p+1) = \sum_{j=0}^{p-1} a_{F(j-id)}\gamma^j$.

Applying Equations (16) and (17), the proposed LSD-first scalable systolic GNB multiplication is addressed as follows:

Algorithm 4. (LSDGNB scalable multiplication)
Input: $A=(a_0,a_1,\cdots,a_{m-1})$ and $B=(b_0,b_1,\cdots,b_{m-1})$ are two normal basis elements in GF(2^m)
Output: $C=(c_0,c_1,\cdots,c_{m-1})=AB$
1. Initial step:
1.1 $A=(a_{F(0)},a_{F(1)},\cdots,a_{F(p-1)})\leftarrow(a_0,a_1,\cdots,a_{m-1})$
1.2 $B=(b_{F(0)},b_{F(1)},\cdots,b_{F(p-1)})\leftarrow(b_0,b_1,\cdots,b_{m-1})$
1.3 $B=\sum_{i=0}^{n-1}B_i\gamma^{di}$, where $B_i=\sum_{j=0}^{d-1}b_{F(id+j)}\gamma^j$ and $n=\lceil p/d\rceil$
1.4 $C=0$
2. Multiplication step:
2.1 for $i=0$ to $n-1$ do
2.2 $C=C+PM(A,B_i)$ (where $PM(A,B_i)$ as referred to Algorithm 3)
2.3 $A=A\gamma^d \bmod (\gamma^p+1)$
2.4 endfor
3. Basis conversion step:
3.1 $C=(\underbrace{c_0,\cdots,c_0}_{t},\cdots,c_{m-1},\cdots,c_{m-1})\leftarrow(c_{F(0)},c_{F(1)},\cdots,c_{F(p-1)})$
4. Return $(c_0,c_1,\cdots,c_{m-1})$

The proposed LSDGNB scalable multiplication algorithm is split into n-loop partial multiplications. Figure 4 depicts the LSDGNB multiplier based on the proposed partial multiplier in Fig.3. Both NB elements A and B are initially transformed into the redundant basis given by Equation (4), and are stored in both registers A and B, respectively. In round 0 (see Figure 4), the systolic array in Figure 3 is adopted to compute $C=A^{(0)}B_0$, and the result is stored in register C. In round 1, the element A must be cyclically shifted to the right by d digits. The result produced in the systolic array is added to register C in the round 0. The first round, which estimates the latency, requires $d+n$ clock cycles. Each subsequent round computation requires a latency of $n+1$ clock cycles. Finally, the entire multiplication requires a latency of $d+n(n+1)$ clock cycles. The critical propagation delay of every cell is the total delay of one 2-input AND gate, one 2-input XOR gate and one 1-bit latch.

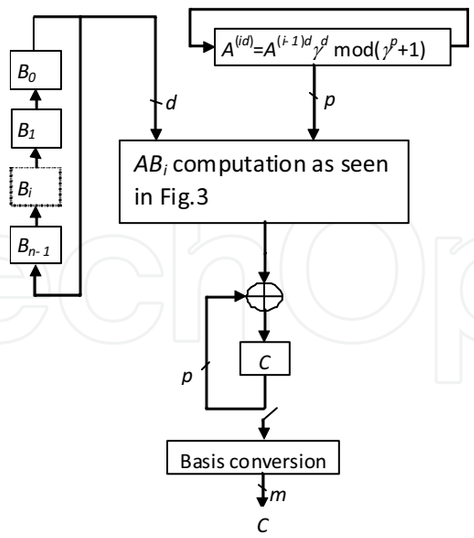

Fig. 4. The proposed LSD-first scalable systolic GNB multiplier over GF(2^m)

3.2 MSD-first scalable multiplier

From Equation (17), the product C can also be re-written by

$$C=(\ldots(AB_{n-1} \bmod (\gamma^p+1))\ \gamma^d+AB_{n-2} \bmod (\gamma^p+1)\ )\ \gamma^d+\ldots)\ \gamma^d+AB_0 \bmod (\gamma^p+1) \quad (18)$$

Therefore, the MSD-first scalable systolic multiplication is addressed using the following algorithm.

Algorithm 5. (MSDGNB scalable multiplication)

Input: A and B are two normal basis elements in GF(2^m)

Output: $C=AB$

1. initial step:

1.1 $A=(a_{F(0)},a_{F(1)},\cdots,a_{F(p-1)}) \leftarrow (a_0,a_1,\cdots,a_{m-1})$

1.2 $B=(b_{F(0)},b_{F(1)},\cdots,b_{F(p-1)}) \leftarrow (b_0,b_1,\cdots,b_{m-1})$

1.3 $B=\sum_{i=0}^{n-1} B_i\gamma^{di}$, where $n=\lceil p/d \rceil$ and $B_i=\sum_{j=0}^{d-1} b_{F(id+j)}\gamma^j$

1.4 $C=0$

2. multiplication step:

2.1 for $i=1$ to n do

2.2 $C=C\gamma^d \bmod (\gamma^p+1)+PM(A,B_{n-i})$, where $PM(A,B_{n-i})$ as referred to Algorithm 3

2.3 endfor

3. basis conversion step:

3.1 $C=(\underbrace{c_0,\cdots,c_0}_{t},\cdots,c_{m-1},\cdots,c_{m-1}) \leftarrow C=(c_{F(0)},c_{F(1)},\cdots,c_{F(p-1)})$

4. return $(c_0,c_1,\cdots,c_{m-1})$

Algorithm 5 presents the MSD-first scalable multiplication, and Figure 5 presents the entire GNB multiplier architecture. As compared to both GNB multiplier architectures, the LSDGNB multiplier before each round computation, the element A must be performed by $A^{(id)}=A^{(i-1)d}\gamma^d \bmod(\gamma^p+1)$. The MSDGNB multiplier after each round computation, the result C

must be performed by $C^{(id)}=C^{(i-1)d}\gamma^{d} \bmod(\gamma^{p}+1)$. Notably, both operations $A^{(id)}$ and $C^{(id)}$ represent a right cyclic shift to id positions. Hence, the two proposed architectures have the same time and space complexity.

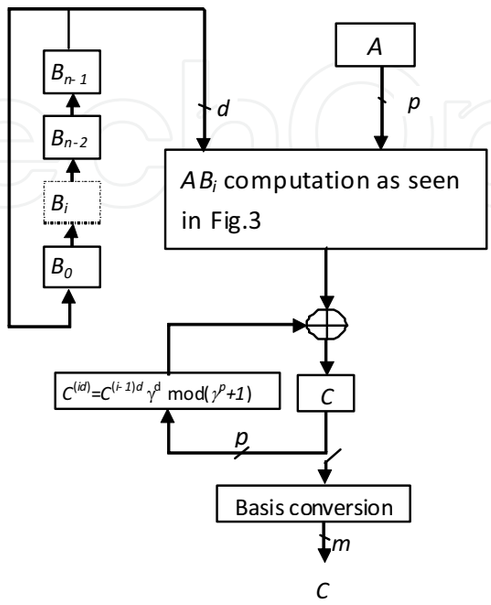

Fig. 5. The proposed MSD-first scalable systolic GNB multiplier over GF(2^m)

4. Modified Scalable and systolic GNB multiplier over GF(2^m)

In the previous section, two scalable GNB multipliers are including one $d\times d$ Hankel multiplier, four registers and one final reduction polynomial circuit. For the whole multiplication scheme, both circuits demand $n(n+1)$ Hankel multiplications. To reduce time- and space-complexity, this section will develop another version of the GNB scalable multiplication scheme.

Let A and B in GF(2^m) be represented by the redundant basis representation. If the selected digital size is d digits, then element B can be represented as $B=\sum_{i=0}^{n-1} B_i\gamma^{di}$, where $B_i=\sum_{j=0}^{d-1} b_{F(id+j)}\gamma^{j}$ and $n=\lceil (mt+1)/d \rceil$. From Equation (14), using LSD-first multiplication algorithm, the partial product can be modified by

$$\begin{aligned} C_0 &= AB_0 \bmod(\gamma^{p}+1) \\ &= Ab_{F(0)}+Ab_{F(1)}\gamma+\ldots+Ab_{F(d-1)}\gamma^{d-1} \bmod(\gamma^{p}+1) \\ &= A^{(0)}b_{F(0)}+A^{(1)}b_{F(1)}+\ldots+A^{(d-1)}b_{F(d-1)} \\ &= c_{0,F(0)}+c_{0,F(1)}\gamma+\ldots+c_{0,F(p-1)}\gamma^{p-1} \end{aligned} \tag{19}$$

where, $A^{(j)} = A\gamma^j \bmod(\gamma^p+1) = \sum_{i=0}^{p-1} a_{F(i-j)}\gamma^j$, for $0 \le j \le d-1$ and $c_{0,F(i)} = \sum_{j=0}^{d-1} b_{F(j)}a_{F(i-j)}$, for $0 \le i \le p-1$.

In (Wu, Hasan, Blake & Gao, 2002), it is shown that, from the GNB representation in (Reyhani-Masoleh & Hasan, 2005) to convert the normal basis, the minimum representation of A has a Hamming weight equal to or less than $mt/2$ if m is even, and $(mt-t+2)/2$ if m is odd. Assume that coordinate numbers of the partial product C_0 in Equation (19) are selected by $q=dk$ consecutive coordinates to satisfy the corresponding normal basis representation, where $q \ge mt/2$ if m is even, and $q \ge (mt-t+2)/2$ if m is odd. Then, the partial product AB_0 can be calculated by

$AB_0 = h_{k-1}B_0+h_{k-2}B_0\gamma^d +\ldots+h_0B_0\gamma^{d(k-1)}$.

Similarly,

$AB_1\gamma^d = h_kB_1+h_{k-1}B_1\gamma^d +\ldots+ h_1B_1\gamma^{d(k-1)}$

$AB_2\gamma 2^d = h_{k+1}B_2+h_kB_2\gamma^d +\ldots+ h_2B_2\gamma^{d(k-1)}$

$\vdots$

$AB_{n-1}\gamma^{(n-1)d} = h_{k+n-2}B_{n-1}+h_{k+n-3}B_{n-1}\gamma^d +\ldots+ h_{n-1}B_{n-1}\gamma^{d(k-1)}$

Thus, the modified multiplication requires only nk Hankel multiplication. As stated above, the modified LSD-first scalable multiplication is addressed as follows:

Algorithm 6. (modified LSDGNB scalable multiplication)

Input: $A=(a_0,a_1,\cdots,a_{m-1})$ and $B=(b_0,b_1,\cdots,b_{m-1})$ are two normal basis elements in GF(2^m)

Output: C= $(c_0,c_1,\cdots,c_{m-1})$ =AB

1. initial step:

1.1 $A=(a_{F(0)},a_{F(1)},\cdots,a_{F(p-1)}) \leftarrow (a_0,a_1,\cdots,a_{m-1})$

1.2 $B=(b_{F(0)},b_{F(1)},\cdots,b_{F(p-1)}) \leftarrow (b_0,b_1,\cdots,b_{m-1})$

1.3 $B=\sum_{i=0}^{n-1} B_i\gamma^{di}$, where $n=p/d$ and $B_i = \sum_{j=0}^{d-1} b_{F(id+j)}\gamma^j$

1.4 $C=\sum_{i=0}^{k-1} C_i\gamma^{di} = 0$, where $k=q/d$ and $C_i = \sum_{j=0}^{d-1} c_{F(id+j)}\gamma^j$

1.5 All Hankel vector hi's for $0 \le i \le n+k-1$ are converted from the redundant basis representation of A.

2. multiplication step:

2.1 for i=0 to $k-1$ do

2.2 for j=0 to $n-1$ do

2.3 $C_{k-1-i}=C_{k-1-i}+H_{i+j}B_j$

2.4 endfor

2.5 endfor

3. basis conversion step:

3.1 $C=(c_0,\cdots,c_0,\cdots,c_{m-1},\cdots,c_{m-1}) \leftarrow (c_{F(0)},c_{F(1)},\cdots,c_{F(q-1)})$

4. return $(c_0,c_1,\cdots,c_{m-1})$

Applying Algorithm 6, Figure 6 shows a LSDGNB multiplier using the redundant representation. The circuit includes two registers, one $d \times d$ Hankel multiplier, and one

summation circuit. In the initial step, two registers H and B are converted by Steps 1.5 and 1.3, respectively. As for the register Hi represent $(2d–1)$-bit latches; and the register B_i is d-bit latches. The operation of a d × d Hankel matrix-vector multiplier has been described in the previous section. In Figure 6, the MUX is responsible for shifting register H. The SW is in charge of shifting the outcome of the GNB multiplication. As mentioned above, the total Hankel multiplications can be reduced from $n(n+1)$ to nk, where $k=\lceil mt/2d \rceil$ if m is even and $k=\lceil (mt-t+2)/2d \rceil$ if m is odd. By the configuration of Figure 6, the modified multiplier is without the final reduction polynomial circuit. It is revealed that the modified multiplier has lower time- and space-complexity as compared to Figures 4 and 5.

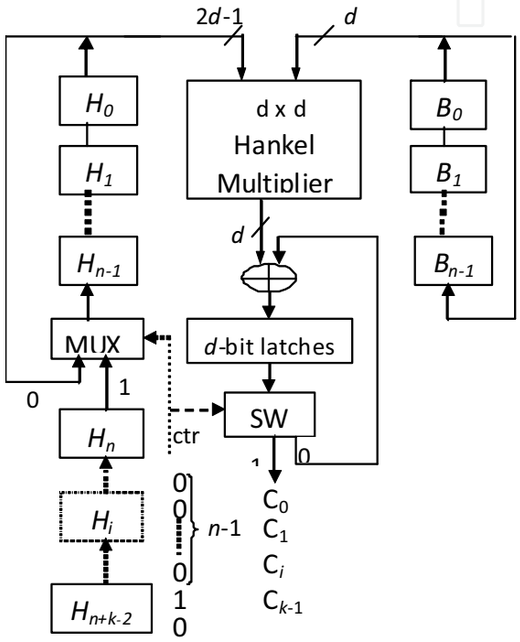

Fig. 6. The modified LSD-first scalable systolic GNB multiplier over GF(2^m)

5. Time and Area Complexity

Various bit-parallel systolic NB multipliers are only discussed on type-1 and 2 ONBs of GF(2^m), as in (Lee & Chiou, 2005 ; Kwon, 2005 ; Lee, Lu & Lee, 2001). As is well known, a type-1 ONB is built from an irreducible AOP, while a type-2 ONB can be constructed from a palindromic representation of polynomials of length $2m$. However, both ONB types exist about 24.5% for m< 1000, as depicted in IEEE Standard P1363 (2000). For the ECDSA (Elliptic Curve Digital Signature Algorithm) applications, NIST (2000) has recommended five binary fields. They are GF(2^{163}), GF(2^{233}), GF(2^{283}), GF(2^{409}) and GF(2^{571}). Their ONB multipliers are implemented with unscalable architectures, and the latency needs $m+1$ clock cycles. The space complexity of the previous architectures is proportional to m^2. As m becomes large, their hardware implements need very large area. Hence, the NIST architectures have limited very applications in cryptography. However, the proposed LSDGNB and MSDGNB multipliers do not have this problem.

Table 2 compares the circuits of the proposed scalable multipliers with those of the other un-scalable (bit-parallel) multipliers. Table 3 lists the proposed multipliers with the total latency. According to this table, the proposed multipliers for a type-2 GNB save about 40% latency as compared to Kwon's (2003) and Lee & Chiou's (2005) multipliers, and those for a type-1 GNB save about 60% latency as compared to Lee-Lu-Lee's multipliers (2001). Since the selected digital size d must minimize the total latency, the proposed multipliers then have low hardware complexity and low latency.

Multipliers	Kwon (2003)	Lee-Lu-Lee (2001)	Lee-Chiou (2005)	LSD-GNBM in Figure 4	MSD-GNBM in Figure 5
Basis	Type-II ONB	Type-I ONB	Type-II ONB	Gaussian NB	Gaussian NB
Architecture	bit-parallel	bit-parallel	bit-parallel	scalable	scalable
Total Complexity 2-input XOR 2-input AND 1-bit latch 1x2 SW	 $2m^2+m$ $2m^2+m$ $5m^2$ 0	 m^2+2m+1 m^2+2m+1 $3m^2+6m+1$ 0	 $2m^2+m$ $2m^2$ $7m^2$	 d^2+d+p d^2 $3.5d^2+5p+3nd$ p	 d^2+d+p d^2 $3.5d^2+5p+3nd$ p
Computation time per cell	T_A+2T_X	T_A+T_X	T_A+T_X	T_A+T_X	T_A+T_X
Latency	m+1	m+1	m+1	d+n(n+1)	d+n(n+1)

Note: the value d is the selected digital size; p=mt+1 is a prime number; $n=\lceil p/d \rceil$; T_X denotes 2-input XOR gate delay; T_A denotes 2-input AND gate delay

Table 2. Comparison of various systolic normal basis multipliers of GF(2^m)

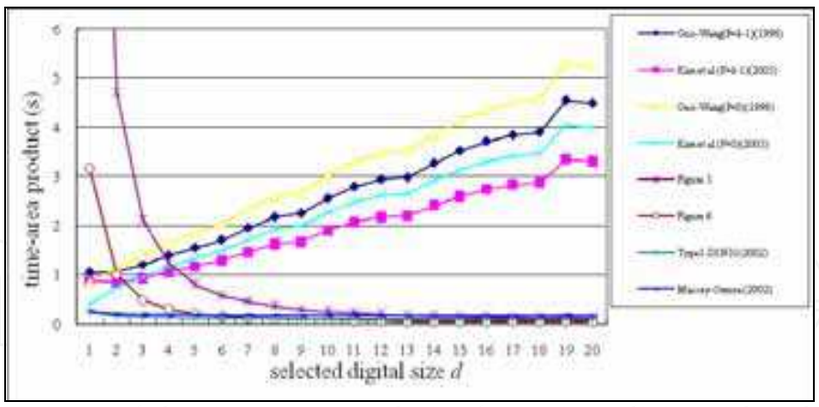

Fig. 7. Comparisons of the time-area complexity for various digit-serial multipliers over GF(2^{233})

Type-2 GNB					Type-1 GNB				
	the proposed architecture		Kwon (2003)	reduced latency		the proposed architecture		Lee-Lu-Lee (2001)	reduced latency
m	*d*	Minimum latency	latency	Compared to Kwon (2003)	*m*	*d*	Minimum latency	latency	Compared to Lee-Lu-Lee (2001)
146	46	88	147	40%	100	29	41	101	59.40%
155	57	87	156	44%	106	31	43	107	59.80%
158	58	88	159	45%	130	30	50	131	61.80%
173	64	94	174	46%	138	31	51	139	63.80%
174	64	94	175	46%	148	34	54	149	63.80%
179	66	96	180	46.60%	162	37	57	163	65%
183	67	97	184	47%	172	39	59	173	65.90%
186	68	98	187	47.60%	178	40	60	179	66.50%
189	69	99	190	47.90%	180	41	61	181	66.30%
191	59	101	192	47.40%	196	44	64	197	67.50%
194	60	102	195	47.70%	210	48	68	211	67.80%
209	65	107	210	49%	226	51	71	227	68.70%

Table 3. Lists the total latency for type-1 and type-2 GNB multipliers over GF(2^m)

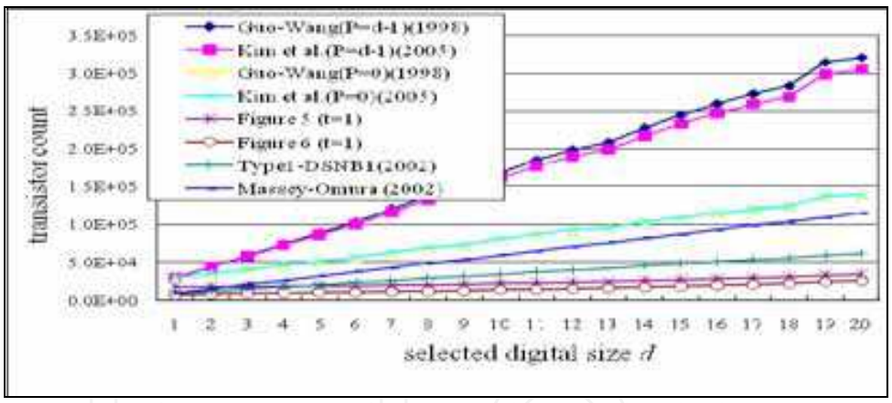

Fig. 8. Comparisons of transistor count for various digit-serial multipliers over GF(2^{233})

By applying the cut-set systolization techniques (Kung, 1988), various digit-serial systolic multipliers are recently reported in (Kim, Hong & Kwon, 2005; Guo & Wang, 1998), which are identical of n processing elements (PE) to enhance the trade-off between throughput performance and hardware complexity. Each PE requires a maximum propagation delay of $T_{max}=(d-1)(T_A+T_{iX}+T_M)+T_A+T_{iX}$, where T_A, T_{iX} and T_M denote the propagation delays through a 2-input AND gate, an i-input XOR gate and a 2-to-1 multiplexer, respectively. The maximum propagation delay in each PE is large if the selected digital size d is large. However, the proposed scalable systolic architectures do not have such problems, since the propagation delay of each PE is independent of the selected digital size d. Applying Horner's rule, Song and Parhi (1998) suggested MSD-first and LSD-first digit-serial multipliers. Various digit-serial multipliers use only one of the input signals A and B to separate $n=\lceil m/d \rceil$ sub-word data. Our

proposed architectures separate both input signals into n sub-word data, in which one of the input element is translated into Hankel vector representation. The proposed LSD-first and MSD-first scalable multiplication algorithms require $n(n+1)$ Hankel multiplications, and the modified multiplication algorithm only demands nk Hankel multiplications, where $k=\lceil mt/2d \rceil$ if m is even and $k=\lceil (mt-t+2)/2d \rceil$ if m is odd. Using a single Hankel multiplier to implement our proposed scalable multipliers, we have $O(d^2)$ space complexity, while other digit-serial multipliers require O(md) space complexity, as seen in Tables 4 and 5.

For comparing the time-area complexity, the transistor count based on the standard CMOS VLSI realization is employed for comparison. Therefore, some basic logic gates: 2-input XOR, 2-input AND, 1×2 SW, MUX and 1-bit latch are assumed to be composed of 6, 6, 6, 6 and 8 transistors, respectively (Kang & Leblebici, 1999). Some real circuits (STMicroelectronics, http://www.st.com) such as M74HC86 (STMicroelectronics, XOR gate, $T_X=12ns$ (TYP.)), M74HC08 (STMicroelectronics, AND gate, $T_A=7ns$ (TYP.)), M74HC279 (STMicroelectronics, SR Latch, $T_L=13ns$ (TYP.)), M74H257 (STMicroelectronics, Mux, $T_M=11ns$ (TYP.)) are employed for comparing time complexity in this paper. In the finite field GF(2^{233}), Figures 7 and 8 show that our proposed scalable multipliers compare to the corresponding digit-serial multipliers (Kim, Hong & Kwon, 2005; Guo & Wang, 1998; Reyhani-Masoleh & Hasan, 2002). As the selected digital size $d \geq 4$, the proposed scalable multipliers have lower time-area complexity than two reported digit-serial PB multipliers (Kim, Hong & Kwon, 2005; Guo& Wang, 1998) (as shown in Figure 7). When the selected digital size $d \geq 8$, the modified scalable multiplier has lower time-area complexity than the corresponding digit-serial NB multiplier (Reyhani-Masoleh & Hasan, 2002). For comparing a transistor count, Figure 8 reveals that our scalable multipliers have low space complexity as compared to the reported digit-serial multipliers (Kim, Hong & Kwon, 2005; Guo & Wang, 1998; Reyhani-Masoleh & Hasan, 2002).

Multipliers	Guo & Wang (1998)	Kim, Hong & Kwon (2005)	Figure 5	Figure 6
Basis	polynomial	polynomial	Gaussian normal	Gaussian normal
Architecture	digit-serial	digit-serial	scalable	scalable
Total Complexity 2-input XOR 2-input AND 1-bit latch 1x2 SW MUX	 $2ed^2$ $e(2d^2+d)$ $\lceil 10d+5Pd \rceil e$ 0 2ed	 $2ed^2$ $e(2d^2+d)$ $\lceil 10d+1+4.5Pd+P \rceil e$ 0 2ed	 d^2+d+p d^2 $3.5d^2+5p+3nd$ p 0	 d^2+d d^2 Q d 1
Critical Path	Pipelined: $d(T_A+2T_X+2T_M)/(P+1)$ Non-Pipelined: $T_A+3T_X+(d-1)(T_A+2T_X+2T_M)$	Pipelined: $d(T_A+T_X+2T_M)/(P+1)$ Non-Pipelined: $T_A+T_X+(d-1)(T_A+T_X+2T_M)$	T_A+T_X	T_A+T_X
Latency	Pipelined: 3+P)e Non-Pipelined: 3e	Pipelined: (3+P)e Non-Pipelined: 3e	d+n(n+1)	d+nk

Note: $k=\lceil h/d \rceil$, $e=\lceil m/d \rceil$; $Q=3.5d^2+nd+(2d-1)(n+k-1)$; TM: denotes 2-by-1 MUX gate delay; P+1 number of pipelining stages inside each basic cell (Kim, Hong & Kwon, 2005; Guo & Wang, 1998)

Table 4. Comparison of various digit-serial systolic multipliers over GF(2m)

Multipliers	Type1-DSNB1 (Reyhani-Masoleh & Hasan , 2002)	Massey-Omura (Reyhani-Masoleh & Hasan , 2002)	Figure 5	Figure 6
Architecture	digit-serial nonsystolic	digit-serial nonsystolic	scalable systolic	scalable systolic
Total Complexity 2-input XOR 2-input AND 1-bit latch 1x2 SW MUX	 (d+1)(m–1) d(m–1)+m 0 0 0	 d(2m–2) d(2m–1) 0 0 0	 d^2+d+p d^2 $3.5d^2+5p+3nd$ p 0	 d^2+d d^2 Q d 1
Critical Path	TA+(1+⌈log2 m⌉)TX	TA+(1+⌈log2 m⌉)TX	TA+TX	TA+TX
Latency	1	1	d+n(n+1)	d+nk

Table 5. Comparison of various digit-serial multipliers for optimal normal basis of GF(2^m)

6. Conclusions

This work presents new multiplication algorithms for the GNB of GF(2^m) to realize LSD-first and MSD-first scalable multipliers. The fundamental difference of our designs from other digit-serial multipliers described in the literature is based on a Hankel matrix-vector representation to achieve scalable multiplication architectures. In the generic field, the GNB multiplication can be decomposed into *n(n+1)* Hankel multiplications. To use the relationship from the GNB to NB, we can modify the LSD-first scalable multiplication algorithm to decrease the number of Hankel multiplication from *n(n+1)* into *nk*, where *k=mt/2d* if m is even and *k=(mt-t-2)/2d* if *m* is odd. Our analysis shows that, in finite field GF(2^{233}), if the selected digital size d ≥ 8, the proposed scalable multipliers then have lower time-area complexity as compared to existing digit-serial multipliers for polynomial basis and normal basis of GF(2^m). Since our proposed scalable multiplication algorithms have highly flexible and are suitable for implementing all-type GNB multiplications. Finally, the proposed architectures have good trade-offs between area and speed for implementing cryptographic schemes in embedded systems.

7. References

Denning, D.E.R.(1983). *Cryptography and Data Security*, Reading, MA: Addison-Wesley.

Rhee, M.Y. (1994). *Cryptography and Secure Communications*, McGraw-Hill, Singapore.

Menezes, A. Oorschot, P. V. & Vanstone, S. (1997). *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL.

Massey, J.L. & Omura, J.K. (1986). Computational method and apparatus for finite field arithmetic," U.S. Patent Number 4,587,627.

Reyhani-Masoleh, A. & Hasan, M.A. (2005). Low complexity word-level sequential normal basis multipliers. *IEEE Transactions on Computers*, Vol. 54, No.2.

Lee, C.Y. & Chang, C.J. (2004). Low-complexity linear array multiplier for normal basis of type-II, *IEEE International Conference on Multimedia and Expo*, Vol. 3, pp. 1515-1518.

Lee, C.Y., Lu, E.H., & Lee, J.Y. (2001). Bit-Parallel Systolic Multipliers for GF(2^m) Fields Defined by All-One and Equally-Spaced Polynomials," *IEEE Transactions on Computers*, Vol. 50, No. 5, pp. 385-393.

Hasan, M.A., Wang, M.Z. & Bhargava, V.K. (1993). A modified Massey-Omura parallel multiplier for a class of finite fields," *IEEE Transactions on Computers*, Vol. 42, No.10, pp. 1278-1280.

Kwon, S. (2003). A low complexity and a low latency bit parallel systolic multiplier over GF(2^m) using an optimal normal basis of type II. *Proceedings of 16th IEEE Symp. Computer Arithmetic*, pp. 196-202.

Lee, C.Y. & Chiou, C.W. (2005). Design of low-complexity bit-parallel systolic Hankel multipliers to implement multiplication in normal and dual bases of GF(2^m). *IEICE Transactions on Fundamentals*, vol. E88-A, no.11, pp. 3169-3179.

IEEE Standard P1363 (2000). *IEEE Standard Specifications for Public-Key Cryptography*.

National Inst. of Standards and Technology,(2000). Digital Signature Standard, FIPS Publication 186-2.

Reyhani-Masoleh, A. (2006). Efficient algorithms and architectures for field multiplication using Gaussian normal bases. *IEEE Transactions on Computers*, Vol. 55, No. 1, pp.34-47.

Lee[1], C.Y. (2003). Low-Latency Bit-Parallel Systolic Multiplier for Irreducible x^m+x^n+1 with gcd(m,n)=1. *IEICE Transactions on Fundamentals*, Vol. E86-A, No.11, pp. 2844-2852.

Lee, C.Y., Horng, J.S. & Jou, I.C. (2005). Low-complexity bit-parallel systolic Montgomery multipliers for special classes of GF(2^m). *IEEE Transactions on Computers*, vol. 54, no.9, pp. 1061-1070.

Lee, C.Y. (2005). Systolic architectures for computing exponentiation and multiplication over GF(2^m) using polynomial ring basis. *Journal of LungHwa University*, vol. 19, pp.87-98.

Lee[2], C.Y. (2003). Low complexity bit-parallel systolic multiplier over GF(2^m) using irreducible trinomials. *IEE Proceeding Computer, and Digital Technical*, Vol. 150, pp. 39-42.

Paar, C., Fleischmann, P. & Soria-Rodriguez, P. (1999). Fast arithmetic for public-key algorithms in Galois fields with composite exponents. *IEEE Transactions on Computers*, vol. 48, no.10, pp. 1025-1034.

Kim, N.Y. & Yoo, K.Y. (2005). Digit-serial AB2 systolic architecture in GF(2^m). *IEE Proceeding Circuits Devices Systems*, Vol. 152, No. 6, pp. 608-614.

Kang, S.M. & Leblebici, Y. (1999). *CMOS Digital Integrated Circuits Analysis and Design*, McGrawHill.

Logic selection guide: *STMicroelectronics*. <http://www.st.com > .

Kim, C.H., Hong, C.P. & Kwon, S. (2005). A digit-serial multiplier for finite field GF(2^m). *IEEE Transactions on VLSI*, Vol. 13, No. 4, pp. 476-483.

Guo, J.H. & Wang, C.L. (1998). Digit-serial systolic multiplier for finite fields GF(2^m). IEE Proc.-Comput. Digit. Tech., Vol. 145, No. 2, pp. 143-148, March.

Kung, S.Y. (1988). *VLSI array processors*, Englewood Cliffs, NJ: Prentice-Hall.

H. Wu, M.A. Hasan, I.F. Blake and S. Gao, "Finite field multiplier using redundant representation. *IEEE Transactions on Computers*, Vol. 51, No. 11, pp.1306-1316, Nov. 2002.

Mullin, R.C., Onyszchuk, I.M., Vanstone, S.A. & Wilson, R. M. (1988/1989). Optimal Normal Bases in GF(p^n). *Discrete Applied Math.*, vol. 22, pp.149-161.

Reyhani-Masoleh, A. & Hasan, M.A. (2003). Fast normal basis multiplication using general purpose processors. *IEEE Transactions on Computers*, Vol. 52, No. 11, pp. 1379-1390.

Song, L. & Parhi, K.K. (1998). Low-energy digit-serial/parallel finite field multipliers. *Journal of VLSI Signal Processing* , Vol.19, pp.149-166.

Tenca, A.F. & Koc, C.K. (1999). A scalable architecture for Montgomery multiplication. *Proceedings of Cryptographic Hardware and Embedded System* (CHES 1999), No. 1717 in Lecture Notes in Computer Science, pp. 94-108.

Reyhani-Masoleh, A. & Hasan, M.A. (2002). Efficient digit-serial normal basis multipliers over GF(2^M). *IEEE International Conference on Circuits and Systems*.

VLSI

Edited by Zhongfeng Wang

ISBN 978-953-307-049-0

Hard cover, 456 pages

Publisher InTech

Published online 01, February, 2010

Published in print edition February, 2010

The process of Integrated Circuits (IC) started its era of VLSI (Very Large Scale Integration) in 1970's when thousands of transistors were integrated into one single chip. Nowadays we are able to integrate more than a billion transistors on a single chip. However, the term "VLSI" is still being used, though there was some effort to coin a new term ULSI (Ultra-Large Scale Integration) for fine distinctions many years ago. VLSI technology has brought tremendous benefits to our everyday life since its occurrence. VLSI circuits are used everywhere, real applications include microprocessors in a personal computer or workstation, chips in a graphic card, digital camera or camcorder, chips in a cell phone or a portable computing device, and embedded processors in an automobile, et al. VLSI covers many phases of design and fabrication of integrated circuits. For a commercial chip design, it involves system definition, VLSI architecture design and optimization, RTL (register transfer language) coding, (pre- and post-synthesis) simulation and verification, synthesis, place and route, timing analyses and timing closure, and multi-step semiconductor device fabrication including wafer processing, die preparation, IC packaging and testing, et al. As the process technology scales down, hundreds or even thousands of millions of transistors are integrated into one single chip. Hence, more and more complicated systems can be integrated into a single chip, the so-called System-on-chip (SoC), which brings to VLSI engineers ever increasingly challenges to master techniques in various phases of VLSI design. For modern SoC design, practical applications are usually speed hungry. For instance, Ethernet standard has evolved from 10Mbps to 10Gbps. Now the specification for 100Mbps Ethernet is on the way. On the other hand, with the popularity of wireless and portable computing devices, low power consumption has become extremely critical. To meet these contradicting requirements, VLSI designers have to perform optimizations at all levels of design. This book is intended to cover a wide range of VLSI design topics. The book can be roughly partitioned into four parts. Part I is mainly focused on algorithmic level and architectural level VLSI design and optimization for image and video signal processing systems. Part II addresses VLSI design optimizations for cryptography and error correction coding. Part III discusses general SoC design techniques as well as other application-specific VLSI design optimizations. The last part will cover generic nano-scale circuit-level design techniques.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Chiou-Yng Lee (2010). Scalable and Systolic Gaussian Normal Basis Multipliers over GF(2m) Using Hankel Matrix-Vector Representation, VLSI, Zhongfeng Wang (Ed.), ISBN: 978-953-307-049-0, InTech, Available from: http://www.intechopen.com/books/vlsi/scalable-and-systolic-gaussian-normal-basis-multipliers-over-gf-2m-using-hankel-matrix-vector-repres

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen

IntechOpen