Low-Latency Digit-Serial and Digit-Parallel Systolic Multipliers for Large Binary Extension Fields

Jeng-Shyang Pan, Senior Member, IEEE, Chiou-Yng Lee, Senior Member, IEEE, and Pramod Kumar Meher, Senior Member, IEEE

Abstract—For cryptographic applications, such as elliptic curve digital signature algorithm (ECDSA) and pairing algorithm, the crypto-processors are required to perform large number of additions and multiplications over finite fields of large orders. To have a balanced trade-off between space complexity and time complexity, in this paper, novel digit-serial and digit-parallel systolic structures are presented for computing multiplication over $GF(2^m)$. Based on novel decomposition algorithm, we have derived an efficient digit-serial systolic architecture, which involves latency of $O((\sqrt{m})^2)$ clock cycles, while the existing digit-serial systolic multipliers involve at least $O(\sqrt{m})$ latency for digit-size $d$. The proposed digit-serial design could be used for AESP-based fields with the same digit-size as the case of trinomial-based fields with a small increase in area. We have also proposed digit-parallel systolic architecture employing $n$-term Karatsuba-like method, where the latency can be reduced from $O(\sqrt{d})$ to $O(\sqrt{m})$. This feature would be a major advantage for implementing multiplication for the fields of large orders. From synthesis results, it is shown that the proposed architectures have significantly lower time complexity, lower area-delay product, and higher bit-throughput than the existing digit-serial multipliers.

Index Terms—Karatsuba-like multiplication, elliptic curve digital signature algorithm, least-significant digit first (LSD-first) multiplication, pairing algorithm, almost equally spaced polynomial (AESP).

I. INTRODUCTION

Finite field arithmetic is widely used in cryptography and error control coding [1], [2]. For cryptographic applications, such as elliptic curve digital signature algorithm (ECDSA) [3], [4], elliptic curve cryptography (ECC) has been used in many security-sensitive applications in various contexts. Recently, cryptographic pairing has been used extensively to derive various security protocols, such as identity based cryptography [5] and short signature scheme [6]. For such protocols, the Weil and Tate pairings based on elliptic curve arithmetic require thousands of additions and multiplication over large finite fields, and have drawn the attention of many researchers [30], [31], [32]. From the point view of VLSI implementations, the realization of pairing computation in resource-constrained applications is highly challenging due to its high computational demand compared to the classical ECC-based crypto-systems. On one hand, the latency of implementations and the logic complexity of arithmetic operations increase with the field order, on the other hand, the number of arithmetic operations increases in this case. For the high-speed architectures, the area and power complexities of implementations of pairing becomes too high. Therefore, based on the requirement of the application, a trade-off is needed to be reached between speed performance and power/energy complexity.

Systolic designs have several advantages such as regularity and modularity of design, simplicity of their processing elements (PE), local interconnections and high-throughput rates. Therefore, several systolic multipliers have been proposed for finite field multipliers. Systolic multipliers for binary extension fields are mainly of two types. Those are bit-parallel and bit-serial multipliers. The systolic multipliers over extended binary field $GF(2^m)$ [7], [8], [11], [21], [25] usually employ either the least-significant bit first (LSB-first) or the most-significant bit first (MSB-first) algorithms. The bit-serial multipliers require less hardware and less power but they are slow. Bit parallel multipliers are fast but involve high hardware and power complexities. To have systolic multipliers with less hardware complexity, the field is usually selected by special polynomials, such as all-one polynomials, pentanomials, and trinomials. Fully bit-parallel systolic multipliers based on Toeplitz matrix-vector product approach are proposed in [9] and [10]. However, these architectures for polynomial basis of $GF(2^m)$ require $O(m^2)$ XOR gates, $O(m^2)$ AND gates, $O(m^2)$ 1-bit latches, and $O(m)$ latency. Recently, by using the special properties of reduction polynomial for trinomials, a super systolic multiplier [22] is proposed to reduce the latency from $O(m)$ to $O(\sqrt{m})$ clock cycles. Bit-serial systolic array multipliers on the other hand require only $O(m)$ space complexity, but they involve longer computational delays.

To have a trade-off between speed and area complexities, digit-serial multipliers have been proposed in the literature. The design of digit-serial multipliers are classified into digit-in-digit-out (DIDO) design, digit-in-parallel-out (DPO) design, and scalable design. The digit-serial polynomial basis multiplier with the DPO structure is proposed in [13], [14] and [23]. A scalable and systolic multiplier using a fixed $d \times d$ bit-parallel Hankel matrix-vector multiplier has been proposed in [15] and [12] whose latency is $(d + [\frac{d}{2}]) ([\frac{d}{2}] - 1)$ clock cycles. Digit-serial systolic multipliers using DIDO architecture are presented in [16], [17] and [24]. The latency of these digit-serial systolic multipliers is $2 [\frac{d}{2}] - 1$ clock cycles. As mentioned above, complexity of digit-serial systolic
finite field multipliers depends on the selected irreducible polynomials and the chosen basis representation.

In this paper, we present a novel decomposition scheme for digit-serial multiplication, and based on that we have derived a low-latency digit-serial systolic multiplier. The proposed digit-serial systolic architecture achieves a latency as low as \(2\sqrt{m/d}\) clock cycles, where \(d\) is the selected digit-size. Under using a fixed digit-size, say \(d = 4\), our proposed digit-serial systolic multiplier for \(GF(2^{409})\) requires 22 clock cycles, while the existing digit-serial multipliers presented in [13] and [14] need 103 clock cycles. When the selected digit-size is one bit, the latency of the proposed multiplier is the same as that of Meher’s multiplier [22]. The divide and conquer algorithm of Karatsuba-Ofman [19] is used to reduce the space complexity of the multiplier in [27] and [28]. Recently, Montgomery has proposed Karatsuba-like function [20]. Based on that we have proposed a digit-parallel systolic multiplier to achieve the trade-off between time and area complexities in the proposed systolic multipliers for large binary extension fields.

The rest of the paper is organized as follows. Section II briefly reviews the classic LSD-first digit-serial multiplier over \(GF(2^m)\). In Section III, we have proposed the novel digit-serial multiplication algorithm to develop a digit-serial systolic multiplier. We have utilized the Karatsuba-like method to derive a digit-parallel systolic multiplier in Section IV. In Section V, time and space complexities of proposed multipliers and corresponding existing works are presented and compared. Finally, we conclude the paper in Section VI.

II. TRADITIONAL DIGIT-SERIAL MULTIPLIER OVER \(GF(2^m)\)

In this section, we briefly review the digit-serial multiplication algorithm [14]. Let the field \(GF(2^m)\) be constructed from an irreducible polynomial \(F(x)\) of degree \(m\). And let two elements \(A\) and \(B\) be represented by the polynomial basis of \(GF(2^m)\), i.e.,

\[
\begin{align*}
A &= a_0 + a_1x + \cdots + a_{m-1}x^{m-1} \\
B &= b_0 + b_1x + \cdots + b_{m-1}x^{m-1}
\end{align*}
\]

where \(a_i\) and \(b_i\) for \(0 \leq i \leq m-1\) are 0 or 1. Finite field multiplication of two elements \(A\) and \(B\) is given by

\[
C = AB \mod F(x)
\]  

Various schemes have been reported in the literature to achieve low hardware implementation of (1) in the resource-constrained environments. Digit-serial multiplier, introduced by [14], provides a trade-off between speed and area complexities. In the following we discuss the Least Significant Digit (LSD) first multiplication to derive a digit-serial multiplier architecture.

Let \(A\), \(B\), and \(C\) be three elements in \(GF(2^m)\) generated by the irreducible polynomial \(F(x)\). Three elements are presented by polynomial basis representation, where \(C = AB \mod F(x)\). Let us assume that \(q = \lfloor \frac{d}{m} \rfloor\), where \(d\) is the selected digit size. If \(m\) is not a multiple of \(d\), then a field element can be padded by \((qd - m)\)-bit zeros as \(A = (a_0, a_1, \ldots, a_{m-1}, 0, \ldots, 0)\). Accordingly, an element \(A\) can be represented by \(A = \sum_{i=0}^{q-1} A_i x^i\), where \(A_i = a_{id} + a_{id+1}x + \cdots + a_{id+d-1}x^{d-1}\). By using LSD-first multiplication scheme, the product \(C\) can be rewritten as

\[
\begin{align*}
C &= AB \mod F(x) = B(A_0 + A_1x^d + \cdots + A_{q-1}x^{(q-1)d}) \mod F(x) \\
&= (C_0 + C_1 + \cdots + C_{q-1}) \mod F(x) \quad (2)
\end{align*}
\]

where

\[
C_i = B^{(i)} A_i
\]  

(3)

\[
B^{(i)} = Bx^{di} \mod F(x) = x^d B^{(i-1)} \mod F(x),
\]  

(4)

\[
B^{(0)} = B.
\]

As mentioned above, the traditional LSD-first multiplication given by (2) can be described by Algorithm 1. Fig. 1 shows a digit-serial multiplier over \(GF(2^m)\) based on Algorithm 1. It consists of one multiplier core, two registers for two reduction operations \((Bx^d \mod F(x)\) and \(C \mod F(x))\), and one \((m + d)\)-bit adder. The multiplier core computes the term \(A_iB\) of step 3 computation. In the initial step, the register \(<B>\) is initialized by the element \(B\), and the register \(<C>\) is initialized by zero. According to LSD-first multiplication of (2), after \([m/d]\) clock cycles, the register \(<C>\) provides \(C = C_0 + C_1 + \cdots + C_{q-1}\). And the final reduction in Step 6 is performed by for computing \(C = C \mod F(x)\) to obtain the complete multiplication. Thus, the architecture of Fig.1 for the LSD-based digit-serial multiplier requires \([m/d] + 1\) clock cycles.

Algorithm 1 Traditional LSD-first multiplication scheme [14]

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

Output: \(C = AB \mod F(x)\)

1. \(C = 0\);
2. \(A = A_0 + A_1x^d + \cdots + A_{q-1}x^{(q-1)d}\), where \(A_i = \sum_{j=0}^{d-1} a_{id+j}x^j\);
3. For \(i = 0\) to \(q - 1\)
4. \(C = C + A_iB\);
5. \(B = Bx^d \mod F(x)\);
6. endfor
7. \(C = C \mod F(x)\)

III. PROPOSED DIGIT-SERIAL SYSTOLIC MULTIPLIER

In order to derive a new digit-serial systolic multiplier, we briefly review the basic properties of the reduction polynomial. Let \(B = \sum_{i=0}^{m-1} b_i x^i\) be the element in \(GF(2^m)\), and the field \(GF(2^m)\) be constructed from an irreducible polynomial of the form \(F(x) = 1 + x^d + x^m\). Let us represent \(xB \mod F(x)\) as

\[
B^{(1)} = xB \mod F(x)
\]

\[
= \sum_{i=0}^{m-1} b_i x^{i+1} \mod F(x)
\]
The product $\bar{C}$ rewritten as

$$\bar{C} = C_{\bar{v}}x^{d\bar{k}} = \sum_{j=0}^{k-1} A_{i+k+j} x^{id} x^{d\bar{k}j} = \sum_{j=0}^{k-1} A_{i+k+j} B_i^{(jd)}$$  \hspace{1cm} (11)$$

where

$$B_i = x^{kd} \mod F(x) = x^{kd} B_{i-1} \mod F(x)$$

As mentioned above, the product $C$ can also be represented as

$$C = \sum_{i=0}^{p-1} \bar{C}_i \mod F(x)$$  \hspace{1cm} (12)$$

The proposed digit-serial multiplication scheme based on (12) is described in Algorithm 2.

**Algorithm 2** Proposed digit-serial multiplication algorithm

- **Inputs:** $A$ and $B$ are two elements in GF($2^m$).
- **Output:** $C = AB \mod F(x)$.

1. **Initialization step**
   $$\bar{C} = 0.$$  
   $$A = \sum_{i=0}^{q-1} A_i x^{id}, \text{ where } A_i = a_{id} + a_{id+1} x + \cdots + a_{id+d-1} x^{id-1}.$$  

2. **Multiplication step**
   2.1. for $i = 0$ to $p - 1$
   2.2. $D = B$
   2.3. $B = x^{kd} B \mod F(x)$
   2.4. for $j = 0$ to $k - 1$
   2.5. $\bar{C} = \bar{C} + DA_{i+k+j}$
   2.6. $D = x^d D \mod F(x)$
   2.7. endfor
   2.8. endfor

3. **Final reduction polynomial step**

   3.1. $C = \bar{C} \mod F(x)$

The proposed digit-serial systolic multiplier architecture for $k = 4$

![Figure 2](image-url)
PE is comprised of one multiplier core, one reduction module R1, one \((m+d)\)-bit adder and a pair of \((m+d)\)-bit latches. The multiplier core computes the partial products \(A_{\text{in}}B_{\text{in}}\), where \(B_{\text{in}}\) is an \(m\) bit word, and \(A_{\text{in}}\) is a \(d\) bit digit. R1 module performs the reduction \(x^dB_{\text{in}}\) mod \(F(x)\), R3 module performs the reduction \(x^dB_{\text{in}}\) mod \(F(x)\). The RAC module (shown in Fig.2) consists of one reduction module R2, one \(m\)-bit adder and one \(m\)-bit register \(<C>\). It performs the final reduction given by step 3.1 of Algorithm 2.

In the proposed digit-serial systolic multiplier of Fig.2, the register \(<B>\) is initialized by the element \(B\), and the register \(<C>\) is initialized by zeros. It performs the LSF-first multiplication according to the proposed scheme based on (12) to compute the product \(C = C_0 + C_1 + \cdots + C_{p-1}\). In the first clock cycle, the register \(<B>\) is fed from left as the input to the proposed multiplier for computing the partial result \(C_0\). Concurrently the reduction module R3 performs \(B_1 = x^dB\) mod \(F(x)\) and stores the reduced operand in register \(<B>\).

In the next clock cycle, \(B_1\) is used as input to the multiplier for computing the partial result \(C_1\), and the reduction module R3 is used for computing \(B_2 = x^dB_1\) mod \(F(x)\) to store the result in the register \(<B>\), and so on. Each of the partial results, \(C_i\), passes through \(k\) PEs followed by RAC module. The result is stored in register \(<C>\) after \(k+1\) clock cycles. The RAC module finally computes \(C = C + C_1\) mod \(F(x)\). Therefore, the proposed digit-serial systolic multiplier computes the multiplication \(C = AB\) mod \(F(x)\) in \(k+p\) clock cycles.

**Theorem 1.** The proposed digit-serial systolic multiplier (as seen in Fig.2) is composed of \(k\) PEs, one RAC, one register \(<B>\), and one R3 module. The latency of the derived architecture requires at most \(2k\) clock cycles, where \(d\) is the selected digit-size, and \(k = \lceil \sqrt{\frac{d}{2}} \rceil\).

**Proof:** Assume that the digit-size is \(d\) and the multiplication is decomposed into \(q\)-term computations, where \(q = \lceil \frac{m}{d} \rceil\). Given the proposed digit-level systolic architecture in Fig. 2, suppose we have \(k\) PEs and one RAC, where \(q = kp\). Thus, the multiplication can also be segmented into \(p\)-term partial results, i.e., \(C = C_0 + C_1 + \cdots + C_{p-1}\). The sub-product \(C_i\) along with \(A_i\) and \(B_i\) are used as inputs to the PEs of the systolic array multiplier. Based on the feature of fully-pipelined systolic array architectures and as mentioned before, the complete multiplication requires \(k+p\) clock cycles. For a given \(q = kp\), if \(k\) (the number of PEs) is smaller, then \(p\) (the number of digit provided as input to the systolic array) becomes large, and consequently, the latency \((k+p)\) becomes large. To have the low latency for finite field multiplications for large values of \(m\), we need to minimize the latency \(k+p\). Hence, we select \(k = p = \lfloor \sqrt{q} \rfloor\). The proposed multiplier can then have the latency of \(2 \lfloor \sqrt{\frac{d}{2}} \rfloor\) clock cycles.

For clarity of the above discussions regarding the proposed digit-serial systolic multiplier, we use the following example to illustrate the PE operations in different clock cycles.

**Example 1.** Let \(A = \sum_{i=0}^{26} a_i x^i\) and \(B = \sum_{i=0}^{26} b_i x^i\) be two elements in \(GF(2^{27})\) generated by the irreducible polynomial \(F(x)\). Let us assume that the selected digit-size is \(d = 3\). Then, we have \(k = \sqrt{\frac{27}{3}} = 3\). And the element \(A\) is decomposed by \(A = \sum_{i=0}^{8} A_i x^{3i}\), where \(A_i = \sum_{j=0}^{2} a_{3i+j} x^j\). Considering (8), the product \(C\) can be represented as \(C = C_0 + C_1 + C_2\) mod \(F(x)\), where \(C_i = \sum_{j=0}^{2} A_{3i+j} B_j x^j\) and \(B_j = x^jB_{i-1}\) mod \(F(x)\) for \(i = 0, 1, 2\). Table I lists each PE operation in every clock cycle. We note that for this case, the proposed digit-serial systolic multiplier requires \(6\) clock cycles.

**IV. PROPOSED DIGIT-PARALLEL SYSTOLIC MULTIPLIER USING KARATSUBA-LIKE SCHEME**

In this section, we use the Karatsuba-like function to realize the digit-parallel systolic multiplier.

**A. Review of Karatsuba-like function**

The divide-and-conquer algorithm for high-precision multiplication was introduced by Karatsuba and Ofman. For modifying Karatsuba function, the Karatsuba-like formulae is suggested by Montgomery [20]. Here we briefly review the Karatsuba-like function.

In finite field \(GF(2^m)\), each field element \(A\) can be represented as \(A = \sum_{i=0}^{m-1} a_i x^i\), where \(a_i \in GF(2)\). The element \(A\) can also be rewritten as

\[ A = A_L + A_H x \]  

where \(A_L = \sum_{j=0}^{\lfloor m/2 \rfloor - 1} a_{2j} x^j\) and \(A_H = \sum_{j=0}^{\lfloor m/2 \rfloor - 1} a_{2j+1} x^j\).

Therefore, for two-term Karatsuba-like function, \(A = A_L^2 + xA_L A_H^2 + B^2 - xB_H^2\) are two polynomials of degree \(m\), where \(A_L, A_H, B_L, B_H\) are \((m/2)\)-bit term polynomials. The product of \(A\) and \(B\) can be rewritten as

\[ C = AB = (A_L^2 + xA_L^2)(B_L^2 + xB_H^2) \]

\[ = A_L^2 B_L^2 + x(A_L B_H + A_H B_L)^2 + A_H^2 B_H^2 x^2 \]

\[ = A_L^2 B_H^2 (1 + x) + (A_L + A_H)^2 (B_L + B_H)^2 x \]

\[ + A_H^2 B_H^2 (x^2 + x) \]

\[ = C_0^2 (1 + x) + C_1^2 (x^2 + x) + C_2^2 x, \]  

\[ \text{ \hspace{1cm} (14)} \]
simplify the subword representation, let us define

\[ B = x^9 \overline{B}_0 \mod F(x) \]
\[ \overline{B}_{11} = x^9 \overline{B}_0 \mod F(x) \]
\[ C_{11} = A_0B_0 \]

In this section we use two-term Karatsuba-like function to obtain as follows.

\[ \overline{B}_{21} = x^9 \overline{B}_1 \mod F(x) \]
\[ \overline{C}_{21} = A_1 \overline{B}_1 \]
\[ \overline{B}_{22} = x^3 \overline{B}_{11} \mod F(x) \]
\[ C_{22} = C_{11} + A_1 \overline{B}_1 \]

\[ \overline{B}_{31} = x^9 \overline{B}_2 \mod F(x) \]
\[ \overline{C}_{31} = A_2 \overline{B}_2 \]
\[ \overline{B}_{32} = x^3 \overline{B}_{21} \mod F(x) \]
\[ C_{32} = C_{21} + A_4 \overline{B}_{21} \]
\[ \overline{B}_{33} = x^3 \overline{B}_{22} \mod F(x) \]
\[ C_{33} = C_{22} + A_2 \overline{B}_{22} \]
\[ \overline{B}_{41} = x^9 \overline{B}_3 \mod F(x) \]
\[ \overline{C}_{41} = A_3 \overline{B}_3 \]
\[ \overline{B}_{42} = x^9 \overline{B}_{31} \mod F(x) \]
\[ C_{42} = C_{31} + A_7 \overline{B}_{31} \]
\[ \overline{B}_{43} = x^9 \overline{B}_{32} \mod F(x) \]
\[ C_{43} = C_{32} + A_7 \overline{B}_{32} \]
\[ C_0 = \overline{C}_{33} \mod F(x) \]
\[ C_1 = \overline{C}_{43} \mod F(x) \]
\[ C_2 = \overline{C}_{53} \mod F(x) \]

Thus, the product \( C \) is represented as:
\[ C = (A_0 \overline{B}_0)^2(1+x) + (A_1 \overline{B}_1)^2(x^2+x) + (A_2 \overline{B}_2)^2x \mod F(x) \] (16)

Assume that \( d \) is the selected digit size, each subword \( \overline{A}_i \) can be rewritten as \( \overline{A}_i = \sum_{j=0}^{\left\lceil \frac{\pi}{2} \right\rceil -1} \overline{a}_{i,j}x^d \), where \( \overline{a}_{i,j} = \sum_{l=0}^{d-1} \overline{a}_{i,j,l}x^l \). According to Theorem 1, assuming \( k \) to be an integer that satisfies \( k = \left\lceil \sqrt{\frac{27}{5}} \right\rceil \), each partial product \( \overline{A}_i \overline{B}_i \) can be represented by

\[ \overline{A}_i \overline{B}_i = \sum_{j=0}^{\left\lfloor \frac{\pi}{2} \right\rfloor -1} \overline{a}_{i,j} \overline{b}_{i,j} x^d \mod F(x) \]
\[ = \sum_{j=0}^{k-1} \overline{c}_{i,j} x^d \mod F(x) \]
\[ = \left( ((\overline{c}_{i,k-1}) x^{dk} + \overline{c}_{i,k-2}) x^{dk-1} + \cdots x^{d1} + \overline{c}_{i,0} \right) x^d \mod F(x) \] (17)

where

\[ \overline{c}_{i,j} = \sum_{l=0}^{k-1} \overline{a}_{i,j,k+l} \overline{b}_{i,l} x^d \]
\[ = \left( ((\overline{a}_{i,j,k+k-1} \overline{B}_i) x^d + \overline{a}_{i,j,k+k-2} \overline{B}_i) x^{d-1} + \cdots x^{d-1} + \overline{a}_{i,j,k} \overline{B}_i \right) x^d \] (18)

Based on (16), (17) and (18), we can derive the digit-parallel multiplication scheme as stated in Algorithm 3.

Fig.4 shows the proposed digit-parallel systolic array architecture using two-term Karatsuba-like function. It consists of three main parts, e.g., pre-processing unit, subword product
Algorithm 3 Digit-parallel multiplication algorithm based on two-term Karatsuba-like function

Inputs: \( A = A_L^1 + A_H^1 x \) and \( B = B_L^2 + B_H^2 x \) are two elements in \( GF(2^m) \).

Output: \( C = AB \mod F(x) \).

1. \( C_0 = 0 \), \( C_1 = 0 \), \( C_2 = 0 \).
2. \( \overline{A}_0 = A_L \), \( \overline{A}_1 = A_H \), \( \overline{B}_0 = B_L \) and \( \overline{B}_1 = B_H \).
3. for \( j = k - 1 \) to 0

**Initialization step**
4. \( \overline{A}_{0,j} = \overline{A}_{0,j} + \overline{A}_{1,j} \), where \( \overline{A}_{i,j} = (\overline{a}_{i,dkj}, \overline{a}_{i,dkj+1}, \ldots, \overline{a}_{i,dkj+dk-1}) \).
5. \( \overline{B}_2 = \overline{B}_0 + \overline{B}_1 \).

**/subword product computation step**
6. \( \overline{C}_0 = \overline{B}_0 \overline{A}_{0,j} \).
7. \( \overline{C}_1 = \overline{B}_1 \overline{A}_{1,j} \).
8. \( \overline{C}_2 = \overline{B}_2 \overline{A}_{1,j} \).
9. \( C_0 = C_0 \times \overline{C}_0 \).
10. \( C_1 = C_1 \times \overline{C}_1 + C_0 \).
11. \( C_2 = C_2 \times \overline{C}_2 + C_1 \).
12. endfor

**/final polynomial reduction step**
13. \( C = C_0^2 (1 + x) + C_1^2 (x^2 + x) + C_2^2 x \mod F(x) \).

Computation (SPC) unit and post-processing unit. In the pre-processing unit, we use \( GF(2^m) \) adder to realize two addition operations of \( \overline{A}_{2,j} = \overline{A}_{0,j} + \overline{A}_{1,j} \) and \( \overline{B}_2 = \overline{B}_0 + \overline{B}_1 \) corresponding to two steps 4 and 5, respectively. The SPC unit consists of three partial product computation array (PCA) to compute \( \overline{C}_0 = \overline{B}_0 \overline{A}_{0,j}, \overline{C}_1 = \overline{B}_1 \overline{A}_{1,j}, \overline{C}_2 = \overline{B}_2 \overline{A}_{1,j} \). Each PCA is a digit-serial systolic array with \( k \)-modified processing elements (PEs), where \( k = \lceil \sqrt{m/2d} \rceil \). It performs partial product computation according to (18). Fig. 5 shows the detailed circuit of the PE. The post-processing unit consists of three accumulation (AC) modules and one final polynomial reduction (FPR) module. Each AC module performs accumulation of partial products. The FPR module performs step 13 to obtain the final results, as shown in Fig.6.

**Theorem 2.** For finite field \( GF(2^m) \) constructed from irreducible polynomials, the latency of the proposed digit-parallel systolic multiplier with using two-term Karatsuba-like function is \( (2 \lceil \sqrt{m/2d} \rceil + 2) \) clock cycles.

**Proof:** Let \( k \) be a positive integer to satisfy \( k = \lceil \sqrt{m/2d} \rceil \), where \( d \) is the selected digit-size. In the proposed digit-parallel systolic multiplier architecture of Fig.4, the three main parts (pre-processing unit, subword product computation unit, and post-processing unit) requires \( 1, k, \) and \( 2 \) clock cycles, respectively. Thus, computing each sub-product \( \overline{C}_{1,j} \) to store in the AC module requires \( k + 2 \) clock cycles. According to Algorithm 3, the main computation requires \( k \) iterations in the for loop of the multiplications, which demands \( 2k + 1 \) clock cycles. Finally, the final reduction in Step 13 performs the summation of three partial results followed by reduction (i.e. \( C = C_0^2 (1 + x) + C_1^2 (x^2 + x) + C_2^2 x \mod F(x) \)) to obtain the product word. Thus, the one complete multiplication requires \( 2k + 2 \) clock cycles.

As mentioned above, in the proposed digit-parallel systolic multiplier (Fig.4), the subword product computation unit consists of three digit-serial systolic array [2] for computing \( \overline{A}_i \overline{B}_i \) of \( i = 0, 1 \) and 2. Observing the structure of Fig.4, each PCA is to calculate the subword product of two \( \frac{m}{2d} \)-bit polynomials. Since three PCAs in Fig.4 are fully parallelism computations, the latency of the proposed multiplier is at most \( (2 \lceil \sqrt{m/2d} \rceil + 2) \) clock cycles, if the selected digit-size is \( d = 1 \).

In this regarding, we employ the recursions of two-term Karatsuba-like function to derive the proposed digit-serial systolic architecture. The proposed multiplier can have the following properties.

**Theorem 3.** Assume that we use \( n \)-term Karatsuba-like function to construct the digit-parallel systolic multiplier, where \( n = 2^t \); then, the subword product computation unit is required
V. TIME AND SPACE COMPLEXITIES

A. Complexities of digit-serial systolic multiplier

Let us consider the following properties to analyze the time and space complexities of the proposed digit-serial systolic multiplier.

**Remark 1.** Let \( F(x) \) be an irreducible trinomials of the form \( F(x) = 1 + x^d + x^m \). The computation of \( x^d B \mod F(x) \) then requires \( d \) XOR gates and involves one XOR gate delay.

**Remark 2.** Let \( F(x) \) be an irreducible polynomial of the form \( F(x) = 1 + x^i + x^j + x^m \) with \( l_1 \approx \frac{m}{2} \), \( l_2 - l_1 \approx \frac{m}{4} \), and \( l_3 - l_2 \approx \frac{m}{8} \). It is shown in [18] that such type of polynomial exists in \( GF(2^m) \) for \( m > 9 \). This polynomial is called an almost equally spaced pentanomial (AESP). In this case, the computation of \( x^d B \mod F(x) \) requires \( 3d \) XOR gates and involves one XOR gate delay.

**Remark 3 (multiplier core).** Let \( A, B, C \) be represented by polynomials given by \( d, m, m+d \) bits, respectively. Then, the computation of \( BA + C \) by traditional grade-school technique, it requires \( dm \) XOR and \( dm \) AND gates, and involves \( TA + (\log_2(d+1))T_X \) gate-delay time.

**Remark 4 (final polynomial reduction).** Let \( C \) be represented by \( m + d \) bit polynomial. Then, computing \( C \mod F(x) \) has the following time and space complexities.

- If \( F(x) \) is an irreducible trinomial, \( C \mod F(x) \) requires \( 2d \) XOR gates and involves one XOR gate-delay.
- If \( F(x) \) is an irreducible AESP, \( C \mod F(x) \) requires \( 6d \) XOR gates and involves one XOR gate-delay.

As shown in the structure of Fig.2, the proposed multiplier consists of \( k \) PEs, one \( m \)-bit register \( < B > \), one reduction module R3, and one RAC module. Each PE consists of one multiplier core, one reduction module R1, one \( m \)-bit \( GF(2^m) \) adder, and a \( (2m + d) \)-bit register. RAC module is comprised of one reduction module R2, one \( m \) XOR gates, and \( m \)-bit register \( < C > \), where the reduction module R2 performs the final reduction operation of step 3.1 according to Algorithm 2. Based on Remarks 1 and 4, we have estimated the space complexity of our proposed architecture for trinomials and AESPs, and listed in Tables II and III. From these tables, we can find that AESP-based digit-serial multiplier (Fig.2) requires \( (4\sqrt{md}+2d) \) number of more XOR gates compared to the trinomial-based digit-serial multiplier. The multiplier has the latency of \( 2 \left \lfloor \sqrt{m/d} \right \rfloor \) clock cycles if \( d \) is the selected digit-size. In [22], Meher has proposed the 2-D super systolic multiplier for trinomials, having \( 2 \left \lfloor \sqrt{m} \right \rfloor \) clock cycles of latency. When the selected digit-size is one bit, the latency of our proposed multiplier in Fig.2 is the same as that of Meher’s multiplier. In this case, the R1 module in Fig.3 can be reduced to one XOR gate, and duration of clock cycle to \( T_A + T_X + T_L \), where \( T_A \), \( T_X \) and \( T_L \) denote the propagation delays of a 2-input AND gate, a 2-input XOR gate and 1-bit latch, respectively. Therefore, latency of the proposed digit-serial multiplier ranges from \( 2 \left \lfloor \sqrt{m} \right \rfloor \) to \( 2 \left \lfloor \sqrt{m/d} \right \rfloor \) clock cycles, which depends on the selected digit-size \( d \).
B. Complexities of digit-parallel systolic multiplier

**Remark 5.** By using $n$-term Karatsuba Algorithm, in the pre-processing unit, the $GF(2^n)$ adder requires $(n \log_2 3 - n)(\lceil \frac{m}{2n} \rceil + \lceil \sqrt{\frac{md}{n}} \rceil)$ XOR gates and $\log_2 m$ XOR gate delays.

Digit-parallel systolic multiplier of Fig.4 is comprised of three main parts. For simplicity of discussion, let us consider the two-term KA to estimate the time- and space-complexities of proposed digit-parallel systolic multiplier. According to Remark 5, the pre-processing unit involves space-complexity of $(\lceil \frac{m}{2n} \rceil + \lceil \sqrt{\frac{md}{n}} \rceil)$ XOR gates and $m$ 1-bit latches, and requires $T_X + T_L$ gate-delay to complete the computation. It consists of 3 $\lceil \sqrt{\frac{mn}{d}} \rceil$ PE's to construct the subword product computation unit. Each $\mathcal{TE}$ involves space complexity of $d \lceil \frac{m}{2n} \rceil$ AND gates, $d \lceil \sqrt{\frac{md}{n}} \rceil$ XOR gates, and $m + \sqrt{\frac{md}{n}}$ latches. Assuming that the field is constructed from an irreducible trinomial, the FPR module in Fig. 6 has space-complexity of $(7m - 3 \lceil \frac{m}{2n} \rceil + 5)$ XOR gates to perform the computation of (16). The critical-path of the proposed architecture for trinomials is $T_A + (\log_2 (d + 1))T_X + T_L$ gate-delay for completing its computation. The post-processing unit consists of an AC module and an FPR module. The AC module consists of $3\lceil \sqrt{\frac{mn}{d}} \rceil$ XOR gates and $3m$ latches. Assuming that the field is constructed from an irreducible trinomial, the FPR module in Fig. 6 has space-complexity of $(7m - 3 \lceil \frac{m}{2n} \rceil + 5)$ XOR gates to perform the computation of (16).

The critical-path of the proposed architecture for trinomials is $T_A + (\log_2 (d + 1))T_X + T_L$. Based on the above discussion we have calculated the time- and space-complexities of the digit-parallel systolic multiplier for trinomials, and listed in Tables II and III. Similarly, we can estimate the complexity of the proposed architecture based on $n$-term KA.

C. Comparisons

Table II lists the hardware components used by our proposed multipliers and the existing digit-serial multipliers [16], [14], [17]. The latency and critical-path of proposed multipliers are compared with existing multipliers in Table III. From this table we can find that the proposed digit-serial and digit-parallel systolic multipliers have latencies of $2 \lceil \sqrt{\frac{mn}{d}} \rceil$ and $2 \lceil \sqrt{\frac{md}{n}} \rceil + 2$ clock cycles, respectively, while traditional digit-serial non-systolic and systolic multipliers involve latencies of $\lceil m/d \rceil + 1$ and $2 \lceil m/d \rceil$ clock cycles, respectively. We note that as shown in this table proposed AESP-based multiplier has also the same latency as the proposed trinomial-based multiplier. In Table II, we have listed the Bit-Throughput $(BT)$ as a measure of speed performance. The $BT$ for our proposed architectures is more than $\sqrt{dm}$, which depends on the selected digit-size $d$.

The applications of Tate and Weil pairing algorithms involve additions and multiplications of very large finite fields. Therefore, we select the field $GF(2^{1223})$ constructed by the trinomial $x^{1223} + x^{155} + 1$ to estimate critical-path, area complexity, and area-delay product for various digit-serial multipliers. We have used the NanGate’s Library Creator and the 45-nm FreePDK Base Kit from North Carolina State University (NCSU) [26] to synthesize the proposed and the corresponding existing digit-serial multipliers and obtained time and area complexities. From Fig. 7, it is shown that computation time of our proposed architectures is significantly lower than those of the existing multipliers. Therefore, time-complexity of our proposed system is much lower than the existing multipliers. Amongst all the existing digit-serial multipliers, the non-systolic multiplier of [14] has the minimum time-complexity. But as shown in Fig. 7, the proposed multiplier of Fig.2 involves nearly 6 to 27 times less time-complexity compared with those of [14] as digit-size increases from 2 to 32. The time-complexity of proposed digit-parallel systolic multiplier using 8-term KA is 1.6 to 2.6 times less than the proposed digit-serial systolic architecture of Fig.2. It is found that proposed multiplier using KA involves the lowest time-complexity amongst the digit-wise systolic multipliers [14], [16], [17]. As shown in Fig. 8, our proposed architectures have higher area-complexity compared to the existing digit-serial multipliers, but as shown in Fig. 9, proposed architectures involve less area-delay product (ADP) than other digit-serial multipliers [14], [16], [17].

For clarity of comparisons, in Table IV, we have listed area, normalized power consumption per GHz and energy per output bit (EOB) of the proposed and the corresponding existing digit-serial multipliers for digit-size $d = 8$. The estimation of EOB is explained in the following.

- The multiplier in a given clock period computes “L” bits of the product word, which could be considered as bit-throughput (BT).
- The multiplier in a given clock period consumes “E” amount of energy.

We can compute the energy consumed per cycle as $E = (\text{power consumption}) \times (\text{clock period})$. Then the EOB is defined as

$$EOB = \frac{E}{L} = \frac{E}{\text{the number of output bits produced per cycle}} \times (\text{clock period})$$

(19)

The clock period mentioned in expression (19) is the clock period used for estimating the power consumption. As shown in Table IV, the structure of [14] has the lowest critical path among the existing digit-serial designs, is 6.2 times more than the proposed digit-serial structure of Fig.2, and 7.7 times and 11 times of the proposed digit-parallel structures of Fig.4 for 2- and 4-term KAs, respectively. The proposed digit-parallel systolic multiplier using four-term KA can save about 55.2% ADP and 56.04% EOB over the best of the existing digit-serial multipliers [14], [16], [17]. Moreover, the digit-parallel systolic multiplier using four-term KA can save about 36.84% EOB over the digit-parallel systolic multiplier using two-term KA. In Table IV, it is shown that the lattencies of our proposed architectures are lower than those of the existing multipliers. For digit-size $d = 8$, our proposed architectures can have $BT > 94$, while the existing multipliers have $BT \leq 8$. Therefore, the proposed digit-parallel systolic multipliers using KA with different number of terms could be used to have the desired trade-off among speed, ADP/EOB, and BT of digit-wise multipliers for large fields.

VI. Conclusions

We have presented two novel low-latency digit-serial and digit-parallel systolic multipliers over $GF(2^n)$ of large orders. The proposed digit-serial architecture for trinomials and AESPs has latency of $2 \lceil \sqrt{\frac{mn}{d}} \rceil$ clock cycles, which is much
Table II. Comparison of space complexities of multipliers

<table>
<thead>
<tr>
<th>Multipiers</th>
<th>#AND</th>
<th>#XOR</th>
<th>#MUX</th>
<th>#Latch</th>
<th>BT (bits per cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 2 for $x^n + x^m + 1$</td>
<td>$m\sqrt{m}$</td>
<td>$\sqrt{md}(2 + m) + d$</td>
<td>$\sqrt{m}(2m + d − 1) + 2m$</td>
<td>$\sqrt{m}$</td>
<td>$\sqrt{dm}$</td>
</tr>
<tr>
<td>Fig. 2 for AESPs</td>
<td>$m\sqrt{m}$</td>
<td>$\sqrt{md}(6 + m) + 3d$</td>
<td>$\sqrt{m}(2m + d − 1) + 2m$</td>
<td>$\sqrt{dm}$</td>
<td>$\sqrt{dm}$</td>
</tr>
<tr>
<td>Fig. 4 for $x^n + x^m + 1$ (two-term KA)</td>
<td>$1.5m\sqrt{md}$</td>
<td>$8m + (1.5m + 3)\sqrt{md} − \frac{n}{2} + 5$</td>
<td>$4.5m + 1.5m\sqrt{md} + \frac{n}{2}$</td>
<td>$\sqrt{dm}$</td>
<td>$\sqrt{dm}$</td>
</tr>
<tr>
<td>Fig. 4 for $x^n + x^m + 1$ (four-term KA)</td>
<td>$\frac{2m}{5}\sqrt{md}$</td>
<td>$\frac{20m}{9} + \left(\frac{2m}{9} + 4.5\right)\sqrt{md} + n$</td>
<td>$\frac{31m}{5} + \sqrt{2md} + \frac{2m}{\sqrt{d}}$</td>
<td>$\sqrt{md}$</td>
<td>$\sqrt{md}$</td>
</tr>
<tr>
<td>Kumar et al. [14]</td>
<td>$(m + k + 1)d + (k + 1)(d − 1)$</td>
<td>$(m + k + 1)d + (k + 1)(d − 1)$</td>
<td>$2m + d + k$</td>
<td>$\frac{m}{10}m + d + 1$</td>
<td>$d$</td>
</tr>
<tr>
<td>Talapatra et al. [17]</td>
<td>$md + 2d$</td>
<td>$2m$</td>
<td>$4m + 3d + 1$ &amp; 1</td>
<td>$d$</td>
<td></td>
</tr>
<tr>
<td>Kim et al. [16]</td>
<td>$2md + m$</td>
<td>$2md$</td>
<td>$2m + 3md + 2 + \sqrt{sd} + s$</td>
<td>$d$</td>
<td></td>
</tr>
</tbody>
</table>

Note: $s + 1$ is the number of pipelined stages in per basic cell, $d$ is the selected digit-size, $k$ is the second high bit number of the irreducible polynomial

Table IV. Comparison of various digit-serial multipliers over $GF(2^{1223})$ in the terms of latency, ADP, critical path delay $T_{CPD}(ns)$, area ($\mu m^2$), power ($\mu W/GHz$), EOB (pJ), and BT(bit/cycle) for digit-size $d = 8$.

<table>
<thead>
<tr>
<th>Multipiers</th>
<th>latency(cycles)</th>
<th>area ($\mu m^2$)</th>
<th>$T_{CPD}(ns)$</th>
<th>ADP ($\mu m^2$)</th>
<th>Power($\mu W/GHz$)</th>
<th>BT (bits per cycle)</th>
<th>EOB(pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 2</td>
<td>25</td>
<td>383,282.6</td>
<td>0.21</td>
<td>2,012,233.6</td>
<td>629,700.9</td>
<td>94.08</td>
<td>6,693</td>
</tr>
<tr>
<td>Fig. 4 (2-term KA)</td>
<td>18</td>
<td>456,965.1</td>
<td>0.21</td>
<td>1,727,328.1</td>
<td>801,322.9</td>
<td>122.3</td>
<td>6,552</td>
</tr>
<tr>
<td>Fig. 4 (4-term KA)</td>
<td>13</td>
<td>433,796.5</td>
<td>0.21</td>
<td>1,184,264.5</td>
<td>722,910.9</td>
<td>174.7</td>
<td>4,138</td>
</tr>
<tr>
<td>Kumar multiplier [14]</td>
<td>154</td>
<td>41,034.97</td>
<td>0.21</td>
<td>1,424,090.9</td>
<td>74,756.56</td>
<td>7.94</td>
<td>9,413</td>
</tr>
<tr>
<td>Kim multiplier [16]</td>
<td>459</td>
<td>161,433.8</td>
<td>0.47</td>
<td>35,626,910.7</td>
<td>216,886</td>
<td>8</td>
<td>27,081</td>
</tr>
<tr>
<td>Talapatra multiplier [17]</td>
<td>506</td>
<td>52,840.1</td>
<td>0.25</td>
<td>4,082,267.8</td>
<td>81,174.9</td>
<td>8</td>
<td>20,294</td>
</tr>
</tbody>
</table>

Table III. Comparison of time complexities of multipliers

<table>
<thead>
<tr>
<th>Multipiers</th>
<th>Latency</th>
<th>critical path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 2</td>
<td>$2\sqrt{m}$</td>
<td>$T_A + (\log_2(d + 1))T_X + T_L$</td>
</tr>
<tr>
<td>Fig. 4 (two-term KA)</td>
<td>$2\sqrt{md} + 2$</td>
<td>$T_A + (\log_2(d + 1))T_X + T_L$</td>
</tr>
<tr>
<td>Fig. 4 (four-term KA)</td>
<td>$2\sqrt{md} + 2$</td>
<td>$T_A + (\log_2(d + 1))T_X + T_L$</td>
</tr>
<tr>
<td>Kumar et al. [14]</td>
<td>$\left\lceil\frac{md}{d + 1}\right\rceil$</td>
<td>$T_A + (\log_2(d + 1))T_X + T_L$</td>
</tr>
<tr>
<td>Talapatra et al. [17]</td>
<td>$2 \left\lceil\frac{md}{d + 1}\right\rceil$</td>
<td>$T_A + (\log_2(d + 1))T_X + T_MUX + T_L$</td>
</tr>
<tr>
<td>Kim et al. [16]</td>
<td>$3 \left\lceil\frac{md}{d + 1}\right\rceil$</td>
<td>$d(T_A + T_X + T_MUX)/(s + 1) + T_P$</td>
</tr>
</tbody>
</table>

Note: (1) $s + 1$ is the number of pipelined stages in per basic cell, $d$ is the selected digit-size. (2) $T_A$, $T_X$, $T_L$ and $T_MUX$ denote the propagation delays of a 2-input AND gate, a 2-input XOR gate, 1-bit latch and a $2 \times 1$ MUX gate, respectively

Figure 7. Comparison of computation time $(ns)$ for various digit-serial multipliers over $GF(2^{1223})$

Figure 8. Comparison of area complexity for various digit-serial multipliers over $GF(2^{1223})$

like method increases, it provides significantly higher bit-throughput and less critical-path, ADP and EOB. The analytical results provide a valuable reference for implementing pairing algorithm and elliptic curve digital signature algorithm (ECDSA) in resource-constrained embedded systems and smart phones. Moreover, our proposed systolic architectures have the features of regularity, modularity, and concurrency, and are suitable for VLSI chip designs on hardware platforms such as ASIC and FPGA.

REFERENCES

Figure 9. Comparison of area-delay products for various digit-serial multipliers over GF($2^{1223}$)

<table>
<thead>
<tr>
<th>Digit-size</th>
<th>Area-delay product (μm²*ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>640000</td>
</tr>
<tr>
<td>3</td>
<td>1280000</td>
</tr>
<tr>
<td>5</td>
<td>2560000</td>
</tr>
<tr>
<td>7</td>
<td>5120000</td>
</tr>
<tr>
<td>11</td>
<td>2^11</td>
</tr>
<tr>
<td>13</td>
<td>2^13</td>
</tr>
<tr>
<td>15</td>
<td>2^15</td>
</tr>
<tr>
<td>17</td>
<td>2^17</td>
</tr>
<tr>
<td>19</td>
<td>2^19</td>
</tr>
<tr>
<td>21</td>
<td>2^21</td>
</tr>
<tr>
<td>23</td>
<td>2^23</td>
</tr>
<tr>
<td>25</td>
<td>2^25</td>
</tr>
<tr>
<td>27</td>
<td>2^27</td>
</tr>
<tr>
<td>29</td>
<td>2^29</td>
</tr>
<tr>
<td>31</td>
<td>2^31</td>
</tr>
<tr>
<td>33</td>
<td>2^33</td>
</tr>
<tr>
<td>35</td>
<td>2^35</td>
</tr>
<tr>
<td>37</td>
<td>2^37</td>
</tr>
<tr>
<td>39</td>
<td>2^39</td>
</tr>
<tr>
<td>41</td>
<td>2^41</td>
</tr>
<tr>
<td>43</td>
<td>2^43</td>
</tr>
<tr>
<td>45</td>
<td>2^45</td>
</tr>
<tr>
<td>47</td>
<td>2^47</td>
</tr>
<tr>
<td>49</td>
<td>2^49</td>
</tr>
<tr>
<td>51</td>
<td>2^51</td>
</tr>
<tr>
<td>53</td>
<td>2^53</td>
</tr>
<tr>
<td>55</td>
<td>2^55</td>
</tr>
<tr>
<td>57</td>
<td>2^57</td>
</tr>
<tr>
<td>59</td>
<td>2^59</td>
</tr>
<tr>
<td>61</td>
<td>2^61</td>
</tr>
<tr>
<td>63</td>
<td>2^63</td>
</tr>
<tr>
<td>65</td>
<td>2^65</td>
</tr>
<tr>
<td>67</td>
<td>2^67</td>
</tr>
<tr>
<td>69</td>
<td>2^69</td>
</tr>
<tr>
<td>71</td>
<td>2^71</td>
</tr>
<tr>
<td>73</td>
<td>2^73</td>
</tr>
<tr>
<td>75</td>
<td>2^75</td>
</tr>
<tr>
<td>77</td>
<td>2^77</td>
</tr>
<tr>
<td>79</td>
<td>2^79</td>
</tr>
<tr>
<td>81</td>
<td>2^81</td>
</tr>
<tr>
<td>83</td>
<td>2^83</td>
</tr>
<tr>
<td>85</td>
<td>2^85</td>
</tr>
<tr>
<td>87</td>
<td>2^87</td>
</tr>
<tr>
<td>89</td>
<td>2^89</td>
</tr>
<tr>
<td>91</td>
<td>2^91</td>
</tr>
<tr>
<td>93</td>
<td>2^93</td>
</tr>
<tr>
<td>95</td>
<td>2^95</td>
</tr>
<tr>
<td>97</td>
<td>2^97</td>
</tr>
<tr>
<td>99</td>
<td>2^99</td>
</tr>
<tr>
<td>101</td>
<td>2^101</td>
</tr>
<tr>
<td>103</td>
<td>2^103</td>
</tr>
<tr>
<td>105</td>
<td>2^105</td>
</tr>
<tr>
<td>107</td>
<td>2^107</td>
</tr>
<tr>
<td>109</td>
<td>2^109</td>
</tr>
<tr>
<td>111</td>
<td>2^111</td>
</tr>
<tr>
<td>113</td>
<td>2^113</td>
</tr>
<tr>
<td>115</td>
<td>2^115</td>
</tr>
<tr>
<td>117</td>
<td>2^117</td>
</tr>
<tr>
<td>119</td>
<td>2^119</td>
</tr>
<tr>
<td>121</td>
<td>2^121</td>
</tr>
<tr>
<td>123</td>
<td>2^123</td>
</tr>
</tbody>
</table>

Pramod Kumar Meher (SM03) received the M.Sc. degree in physics and the Ph.D. degree in science from Sambalpur University, India, in 1978, and 1996, respectively.

Currently, he is a Senior Scientist with the Institute for Infocomm Research, Singapore, and Adjunct Professor with the School of Electrical Sciences, Indian Institute of Technology Bhubaneswar, India. Previously, he was a Professor of Computer Applications with Utkal University, India, from 1997 to 2002, and a Reader in electronics with Berhampur University, India, from 1993 to 1997. His research interest includes design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal, image and video processing, communication, bio-informatics and intelligent computing. He has contributed more than 200 technical papers to various reputed journals and conference proceedings.

Dr. Meher has served as a speaker for the Distinguished Lecturer Program (DLP) of IEEE Circuits Systems Society during 2011 and 2012 and Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS during 2008 to 2011. Currently, he is serving as Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: REGULAR PAPERS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and Journal of Circuits, Systems, and Signal Processing. Dr. Meher is a Fellow of the Institution of Electronics and Telecommunication Engineers, India. He was the recipient of the Samanta Chandrasekhar Award for excellence in research in engineering and technology for 1999.