The ISC Int'l Journal of Information Security

July 2015, Volume 7, Number 2 (pp. 101–114)

http://www.isecure-journal.org

Efficient Implementation of Low Time Complexity and Pipelined Bit-Parallel Polynomial Basis Multiplier over Binary Finite Fields☆

Bahram Rashidi [1,*], Reza Rezaeian Farashahi [2,3], and Sayed Masoud Sayedi [1]

[1] *Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran*
[2] *Department of Mathematical Sciences, Isfahan University of Technology, Isfahan, Iran*
[3] *School of Mathematics, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 28 January 2015
First Revised: 23 June 2015
Last Revised: 8 August 2015
Accepted: 26 August 2015
Published Online: 2 September 2015

Keywords:
Bit-parallel Multiplier, Elliptic Curve Cryptography, Trinomials, Pentanomials, Pipelining.

ABSTRACT

This paper presents two efficient implementations of fast and pipelined bit-parallel polynomial basis multipliers over GF(2^m) by irreducible pentanomials and trinomials. The architecture of the first multiplier is based on a parallel and independent computation of powers of the polynomial variable. In the second structure only even powers of the polynomial variable are used. The parallel computation provides regular and low-cost structure with low critical path delay. In addition, the pipelining technique is applied to the proposed structures to shorten the critical path and to perform the computation in two clock cycles. The implementations of the proposed methods over the binary extension fields GF(2^{163}) and GF(2^{233}) have been successfully verified and synthesized using Xilinx ISE 11 by Virtex-4, XC4VLX200 FPGA.

1 Introduction

Elliptic curve cryptography (ECC) is a mechanism for implementing public-key cryptography. The approach is built on the arithmetic of elliptic curves over finite fields. ECC offers equivalent security with smaller key sizes, compared to other asymmetric cryptosystems such as RSA. The efficient implementation of ECC is performed by applying efficient computational point multiplication algorithms. Moreover, the curve operations are performed using arithmetic operations in the underlying field. The ECC point multiplication mechanism can be categorized into three main levels. First level is the finite field or Galois field (GF) arithmetic which includes field addition, field multiplication, field squaring, and field inversion. Second level is the elliptic curve group operation, i.e. point addition and point doubling. The third level is the computation of the scalar multiplication or point multiplication algorithm. The field multiplication is the basic and the most important field operation in finite fields and their applications in implementing coding algorithms and cryptosystems.

In recent years, several hardware implementations of the polynomial basis multipliers over binary finite fields are presented [1–45]. In [1] a bit-serial polynomial basis multiplier over binary extension elds is proposed by Reyhani-Masoleh. It generates one bit of the multiplication in each clock cycle with the latency of one cycle. Chiou and Jeng [2] presented two low latency systolic multipliers over GF(2^m) based on general irreducible polynomials and irreducible pentanomials. They used a signal flow graph (SFG) to represent the algorithm of multiplication over GF(2^m). Then, from SFG by suitable cut-set retiming, they presented two low latency systolic structures for mul-

☆ This article is an extended version of an ISCISC'2014 paper.
* Corresponding author.
Email addresses: b.rashidi@ec.iut.ac.ir (B. Rashidi), farashahi@cc.iut.ac.ir (R. Rezaeian Farashahi), m_sayedi@cc.iut.ac.ir (S. M. Sayedi).
ISSN: 2008-2045

tiplications over GF(2^m) based on general irreducible polynomials and pentanomials. In [5] a modification of the original Karatsuba-Ofman algorithm, in order to integrate the modular reduction inside the polynomial multiplication step, is proposed by Cuevas-Farfan *et. al.* The modular reduction is achieved by using parallel linear feedback registers. In [7] a non-redundant Karatsuba-Ofman algorithm (NRKOA) with removing redundancy operations is presented by Chang *et. al.* In [9], Rebeiro and Mukhopadhyay proposed a novel hybrid Karatsuba multiplier which uses both simple and general Karatsuba algorithms. Also, they proposed a design for a masked multiplier based on Karatsuba algorithm which requires fewer number of gates. In [13] two novel low-latency digit-serial and digit-parallel systolic multipliers over GF(2^m) are presented by Pan *et. al.* In [15] Selimis *et. al.* proposed a versatile bit-serial MSB multiplier for GF(2^m) elds that achieves a 50% increase on average in throughput compared to other designs. In [16] a one-dimensional array multiplier for performing multiplication in the finite field GF(2^m) is presented by Chiou *et. al.* A linear feedback shift register is employed in their multiplier and a two-dimensional systolic array version of the multiplier is included in their work. Lee *et. al.*, [17], presented a bit-parallel dual basis multiplier using the modied Booths algorithm. Due to the advantage of the modied Booths algorithm, and to reduce space and time complexities, two bits are processed in parallel. In addition, for realization of the multiplication algorithm, they proposed a multiplexer-based structure. Lee *et. al.* [25], presented time-dependent and time-independent multiplication algorithms over finite field GF(2^m) by employing an interleaved conventional multiplication and a folded technique. In [36] authors presented a low-complexity digit-level serial input parallel output (SIPO) Gaussian normal basis multiplier, an improved digit-level parallel input serial output (PISO) multiplier architecture, and a new hybrid architecture by connecting the output of the digit-level PISO multiplier to the input of the digit-level SIPO multiplier. The hybrid multiplier architecture performs double-multiplication with the same number of clock cycles required for one multiplication. The critical path delay in [36] for bit-parallel structure is logarithmically dependent on the type of the Gaussian normal basis and operands length.

The focus of this paper is to provide a high-speed design and implementation of bit-parallel filed multiplier over binary finite fields in polynomial basis by irreducible trinomials and pentanomials. Therefore, two structures are proposed, the first architecture is based on a parallel and independent computation of powers of the polynomial variable. Moreover, in the second structure only even powers of the polynomial variable are used. The parallel computation provides regular and low-cost structure with low critical path delay. The pipelining technique is applied to the proposed structures to shorten the critical path. The rest of this paper is organized as follows. Section 2 provides the proposed architectures of two pipelined bit-parallel multipliers which have regular and parallel structures with low critical path delays. The pipelining technique is used to speed up the hardware implementations of the structures. The results and comparisons with other architectures are given in Section 3. The paper is concluded in Section 4.

2 Proposed Structures for Bit-Parallel Binary Finite Field Multiplier

Binary finite fields of order $2m$ are the extension fields of GF(2) denoted by GF(2^m). The elements of a binary field can be represented using a polynomial basis. The binary field GF(2^m) is constructed by an irreducible polynomial $P(x)$ over GF(2) of degree m, given by

$$P(x) = x^m + p_{m-1}x^{m-1} + \cdots + p_2x^2 + p_1x^1 + p_0$$

Then, every element of GF(2^m) is represented by a polynomial of degree at most $m-1$ and with coefficients over GF(2), i.e, by a polynomial

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x^1 + a_0$$

The polynomial $A(x)$ is simply given by its coefficients in GF(2) as the m-bit number $[a_{m-1}, a_{m-2}, \cdots, a_2, a_1, a_0]$, that is the binary representation of the corresponding element in GF(2^m).

The addition of two elements of the binary field is the usual addition of the two polynomials over GF(2). That is the addition of their coefficients modulo 2, performed by XOR-ing. The multiplication in GF(2^m) is performed by the usual multiplication of the polynomials over GF(2) and then reduction module the polynomial $P(x)$.

2.1 Proposed Structure of the Bit-parallel Multiplier (Method 1)

Here, we explain how to perform multiplication of two elements of GF(2^m) in more details. Let $A(x), B(x)$ be two binary polynomials of degree at most $m-1$, representing these elements. Then,

$$\begin{aligned} A(x)B(x) &= \left(\sum_{i=0}^{m-1} b_i x^i\right) A(x) \ mod \ P(x) \\ &= (b_{m-1}x^{m-1} + \cdots + b_1x + b_0)A(x) \ mod \ P(x) \\ &= b_{m-1}x^{m-1}A(x) \ mod \ P(x) + \cdots + \\ &\quad b_1xA(x) \ mod \ P(x) + b_0A(x) \ mod \ P(x) \end{aligned}$$

There is a recursive method to compute $x^iA(x)$, for $i = 0$ to $m-1$. Suppose for some positive integer i we already obtained $x^{i-1}B(x)$. Then, to compute

$x^i A(x)$, we write $x^i A(x) = x(x^{i-1}A(x))$. If the leading term of $x^{i-1}A(x)$ has degree at most $m-2$, then the degree of $x^i A(x)$ is at most $m-1$. So, in this case to obtain the m-bit representation of $x^i A(x)$, it is only needed to make the left shift bit operation on the m-bit representation of $x^{i-1}A(x)$ and concatenate with a single bit '0' as the least significant bit. Moreover, if the leading term of $x^{i-1}A(x)$ is x^{m-1}, then we have $x^i A(x) = x^m +$ *lower degree terms*. In this case a reduction mod $P(x)$ is required to be done. Therefore, we compute $x^i A\,(x)\, mod P(x)$ sequentially using $x^k A\,(x)\, mod P(x)$, for $k = 0$ to $i-1$. Figure 1 shows the sequence computation of multiplication by powers of the polynomial variable x using a block multiplication by x.

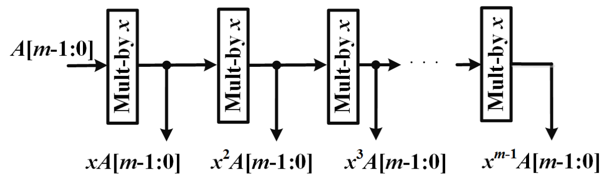

Figure 1. Sequence computation of multiplication by powers of $x$based on multiplication by x.

Algorithm 1 provides the sequential multiplication of two elements of GF(2^m) represented by polynomial $A(x)$ and $B(x)$. In addition, the schematic of this multiplication performed by sequential shift and reduction operations is shown in Figure 2.

Algorithm 1 Sequential shift and reduction multiplication in GF(2^m)

Input: Binary Polynomials $A(x), B(x)$ of degree at most $m-1$.
Output: $C(x) = A(x)B(x)\ mod\ P(x)$

```
M[m-1:0]=A[m-1:0];
R0[m-1:0]=B[0]&M[m-1:0];
for i = 1 to m − 1 do
    if M[m-1]=='1' then
        M[m-1:0]=(M[m-2:0] ||'0') ⊕ P[m-1:0]; // shift and reduction
    else
        M[m-1:0]=M[m-2:0] ||'0'; // shift
    end if
    Ri[m-1:0]=B[i]&M[m-1:0];
end for
C[m-1:0]= R0[m-1:0] ⊕ R1[m-1:0] ⊕ ... Rm−1[m-1:0];
return C[m-1:0];
```

The critical path delay of the structure shown in Figure 2 is very long, since the computations of $x^i A(x)$ are performed sequentially. We propose a structure to implement parallel computation of $A(x)B(x)\ mod\ P(x)$, where $P(x)$ is an irreducible trinomial or pentanomial. In this structure terms $x^i A(x)$, for $i = 0$ to $m-1$, are computed independently and so in parallel. Then the computed terms $x^i A(x)$ after AND-ing with $B(x)$ are added to each other by XOR tree. The schematic of the proposed structure are provided in Figure 3.

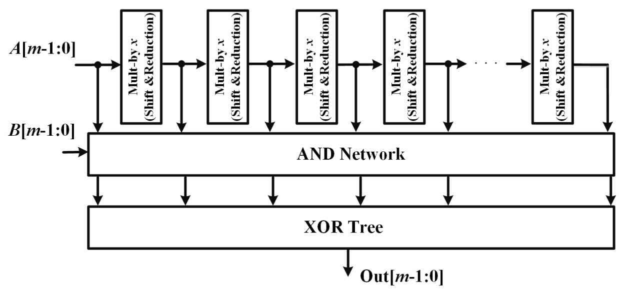

Figure 2. Schematic of the bit-parallel multiplication in GF(2^m) by sequential shift and reduction operations.

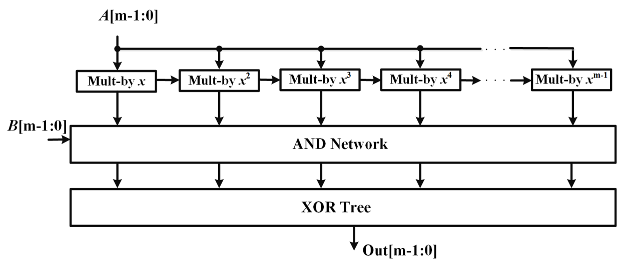

Figure 3. Proposed schematic of the bit-parallel polynomial multiplication.

Here, we explain this method for the trinomial $P(x) = x^m + x^k + 1$. To compute $xA(x)$ bits $[a_{m-1}, a_{m-2}, \ldots, a_2, a_1, a_0]$ of the binary representation of $A(x)$ are shifted one bit to the left as $[a_{m-2}, \ldots, a_2, a_1, a_0, 0]$. If a_{m-1} is '1' the reduction based on $P(x)$ is also required. The reduction by $P(x)$ is performed by

$$[a_{m-2}, a_{m-3}, \ldots, a_{k-1} \oplus 1, \ldots, a_2, a_1, a_0, 0 \oplus 1] = [a_{m-2}, a_{m-3}, \ldots, a_{k-1} \oplus 1, \ldots, a_2, a_1, a_0, 0]$$

Notice the reduction is performed by XOR-ing of bits $[a_{m-2}, a_{m-3}, \ldots, a_k \oplus 1, \ldots, a_2, a_1, a_0, 0]$ with '1' if a_{m-1} is '1'. That is XOR-ing with a_{m-1}. So, the two states of only shift and shift-reduction can be combined in one state by the rotate operation based on the most significant bit a_{m-1}. In other word $xA(x)$is represented by

$$[a_{m-2}, a_{m-3}, \ldots, a_{k-1} \oplus a_{m-1}, \ldots, a_1, a_0, a_{m-1} \oplus 0] = [a_{m-2}, a_{m-3}, \ldots, a_{k-1} \oplus a_{m-1}, \ldots, a_1, a_0, a_{m-1}]$$

This method can compute $x^i A(x)$, for $i = 2$ to $m-1$, recursively. For example, $x^2 A(x)$ is computed by

$$[a_{m-3}, a_{m-4}, \ldots, a_{k-1} \oplus a_{m-1}, a_{k-2} \oplus a_{m-2}, \ldots, a_1, a_0, a_{m-1}, a_{m-2}]$$

and $x^3 A(x)$ given by

$$[a_{m-4}, a_{m-5}, \ldots, a_{k-1} \oplus a_{m-1}, a_{k-2} \oplus a_{m-2}, a_{k-3} \oplus a_{m-3}, \ldots, a_1, a_0, a_{m-1}, a_{m-2}, a_{m-3}]$$

Implementation of $x^i A(x)$, for $i = 0$ to $m-1$ by the proposed method has regular form and is reconfigurable for other irreducible trinomials. Algorithm

2 shows the proposed bit-parallel polynomial basis multiplier over GF(2^m) by the irreducible trinomial $P(x) = x^m + x^k + 1$.

Algorithm 2 Proposed bit-parallel polynomial basis multiplier over GF(2^m) by $P(x) = x^m + x^k + 1$, $1 < k < m/2$

Input: $A = [a_{m-1}, a_{m-2}, \dots, a_1, a_0]$, $B = [b_{m-1}, b_{m-2}, \dots, b_1, b_0]$; where $a_i, b_i \in$ GF(2)
Output: $C = AB \ mod \ P(x)$
1: $M_0[m-1:0] = b_0 \ \& \ [a_{m-1}, a_{m-2}, \dots, a_2, a_1, a_0]$;
2: $M_1[m-1:0] = b_1 \ \& \ [am-2, a_{m-3}, \dots a_{k-1} \oplus a_{m-1}, \dots, a_2, a_1, a_0, a_{m-1}]$;
3: $M_2[m-1:0] = b_2 \ \& \ [am-3, a_{m-4}, \dots a_{k-1} \oplus a_{m-1}, a_{k-2} \oplus a_{m-2}, \dots, a_1, a_0, a_{m-1}, a_{m-2}]$;
4: $M_3[m-1:0] = b_3 \ \& \ [am-4, a_{m-5}, \dots a_{k-1} \oplus a_{m-1}, a_{k-2} \oplus a_{m-2}, a_{k-3} \oplus a_{m-3} \dots, a_1, a_0, a_{m-1}, a_{m-2}, a_{m-3}]$;
5: ...
6: $M_{m-1}[m-1:0] = b_{m-1} \ \& \ [a_0 \oplus a_{m-k}, a_{m-1} \oplus a_{m-k-1}, \dots a_{k+1} \oplus (a_1 \oplus a_{m-k+1}), a_k, a_{k-1} \oplus a_{m-1}, \dots, a_2 \oplus a_{m-k+2}, a_1 \oplus a_{m-k+1}]$;
7: C[m-1:0]= M_0[m-1:0] $\oplus$ M_1[m-1:0] $\oplus \dots \oplus M_{m-1}$[m-1:0];
8: **return** C[m-1:0];

Here, we give an example, over GF(2^{10}). This binary field is constructed by the irreducible trinomial $P(x) = x^{10} + x^3 + 1$. The 10-bit representation of $A(x)$is $[a_9, a_8, a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0]$. Table 1 shows the 10-bit representation of $x^i A(x)$, for $i = 1, \dots, 9$.

Table 1. 10-bit representation $x^i A(x)$

i	$x^i A(x)$
1	$[a_8, a_7, a_6, a_5, a_4, a_3, a_2 \oplus a_9, a_1, a_0, a_9]$
2	$[a_7, a_6, a_5, a_4, a_3, a_2 \oplus a_9, a_1 \oplus a_8, a_0, a_9, a_8]$
3	$[a_6, a_5, a_4, a_3, a_2 \oplus a_9, a_1 \oplus a_8, a_0 \oplus a_7, a_9, a_8, a_7]$
4	$[a_5, a_4, a_3, a_2 \oplus a_9, a_1 \oplus a_8, a_0 \oplus a_7, a_9 \oplus a_6, a_8, a_7, a_6]$
5	$[a_4, a_3, a_2 \oplus a_9, a_1 \oplus a_8, a_0 \oplus a_7, a_9 \oplus a_6, a_8 \oplus a_5, a_7, a_6, a_5]$
6	$[a_3, a_2 \oplus a_9, a_1 \oplus a_8, a_0 \oplus a_7, a_9 \oplus a_6, a_8 \oplus a_5, a_7 \oplus a_4, a_6, a_5, a_4]$
7	$[a_2 \oplus a_9, a_1 \oplus a_8, a_0 \oplus a_7, a_9 \oplus a_6, a_8 \oplus a_5, a_7 \oplus a_4, a_6 \oplus a_3, a_5, a_4, a_3]$
8	$[a_1 \oplus a_8, a_0 \oplus a_7, a_9 \oplus a_6, a_8 \oplus a_5, a_7 \oplus a_4, a_6 \oplus a_3, a_5 \oplus (a_2 \oplus a_9), a_4, a_3, a_2 \oplus a_9]$
9	$[a_0 \oplus a_7, a_9 \oplus a_6, a_8 \oplus a_5, a_7 \oplus a_4, a_6 \oplus a_3, a_5 \oplus (a_2 \oplus a_9), a_4 \oplus (a_2 \oplus a_9), a_3, a_2 \oplus a_9, a_1 \oplus a_8]$

As seen in Table 1 there are many similar terms in computations of $x^i A(x)$. So, resource sharing is used to reduce the hardware of the implementation. Figure 4 shows the proposed architecture of high-speed bit-parallel polynomial basis multiplier over GF(2^{10}) by the irreducible trinomial $P(x) = x^{10} + x^3 + 1$. As seen in Figure 4 two inputs are applied simultaneously to the circuit in parallel form. The critical path of the proposed architecture is $2 + log_2^{10})\ T_X + T_A$, where T_X and T_A are the time delays of a 2-input XOR gate and a 2-input AND gate, respectively. This circuit has low time complexity and the input data are rapidly propagated in path. Furthermore, the final summation is implemented by a tree structure of XOR gates in a parallel form and short path. This structure is well suited to pipelining. Therefore, the critical path is very shorter than that of a sequential summation form. The following Algorithm 3, describes the bit-parallel structure of the polynomial basis multiplier over GF(2^m) constructed by the irreducible pentanomial $P(x) = x^m + x^{k_1} + x^{k_2} + x^{k_3} + 1$. Here,

---

Algorithm 3 Proposed bit-parallel polynomial basis multiplier over GF(2^m) by $P(x) = x^m + x^{k1} + x^{k2} + x^{k3} + 1, 1 < k_3 < k_2 < k_1 < m$:

Input: $A = [a_{m-1}, a_{m-2}, \dots, a_1, a_0] B = [b_{m-1}, b_{m-2}, \dots, b_1, b_0]$; where $a_i, b_i \in GF(2)$
Output: $C = AB \ mod \ P(x)$
1: $M_0[m-1:0] = b_0 \ \& \ [a_{m-1}, a_{m-2}, \dots, a_2, a_1, a_0]$;
2: $M_1[m-1:0] = b_1 \ \& \ [am-2, a_{m-3}, \dots a_{k-1} \oplus a_{m-1}, \dots, a_2, a_1, a_0, a_{m-1}]$;
3: $M_2[m-1:0] = b_2 \ \& \ [am-3, a_{m-4}, \dots a_{k-1} \oplus a_{m-1}, a_{k-2} \oplus a_{m-2}, \dots, a_1, a_0, a_{m-1}, a_{m-2}]$;
4: $M_3[m-1:0] = b_3 \ \& \ [am-4, a_{m-5}, \dots a_{k-1} \oplus a_{m-1}, a_{k-2} \oplus a_{m-2}, a_{k-3} \oplus a_{m-3} \dots, a_1, a_0, a_{m-1}, a_{m-2}, a_{m-3}]$;
5: ...
6: $M_{m-1}[m-1:0] = b_{m-1} \ \& \ [a_0 \oplus a_{m-k}, a_{m-1} \oplus a_{m-k-1}, \dots a_{k+1} \oplus (a_1 \oplus a_{m-k+1}), a_k, a_{k-1} \oplus a_{m-1}, \dots, a_2 \oplus a_{m-k+2}, a_1 \oplus a_{m-k+1}]$;
7: C[m-1:0]= M_0[m-1:0] $\oplus$ M_1[m-1:0] $\oplus \dots \oplus M_{m-1}$[m-1:0];
8: **return** C[m-1:0];

---

we give an example of multiplication over GF(28). This binary field is constructed by the irreducible pentanomial $P(x) = x^8 + x^4 + x^3 + x + 1$. The 8-bit representation of $A(x)$ is $[a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0]$. Table 2 shows the 8-bit representation of $x^i A(x)$, for $i = 1, \dots, 7$. The proposed pipelined architecture of the high-speed bit-parallel multiplier over GF(2^8) by the irreducible pentanomial $P(x) = x^8 + x^4 + x^3 + x + 1$ is shown in Figure 5. The critical data path of the proposed architecture is $(4 + (\lceil \frac{log_2^8}{2} \rceil - 1))T_X + T_A$.

2.2 Proposed Structure of the Bit-parallel Multiplier (Method 2)

Here, we present our second structure of the bit-parallel binary finite field multiplier. The idea of this structure is to separate even and odd degree terms of the corresponding polynomial of one input in the multiplier. Let $A(x)$ and $B(x)$ be two polynomials representing two elements of the binary finite field GF(2^m). Let $C(x)$ be the multiplication of A, B. We write $B(x) = B_{odd}(x) + B_{even}(x)$, where $B_{odd}(x)$ and $B_{even}(x)$ are polynomials including all odd and even degree terms of $B(x)$, respectively. In other words,

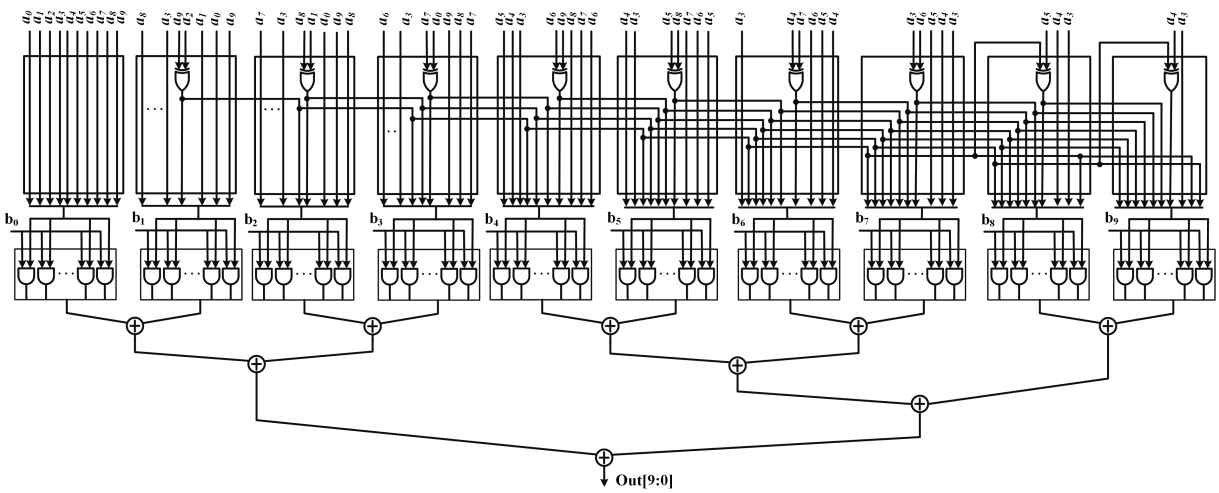

Figure 4. Proposed architecture of the bit-parallel multiplier over GF(2^{10}) by $P(x) = x^{10} + x^3 + 1$.

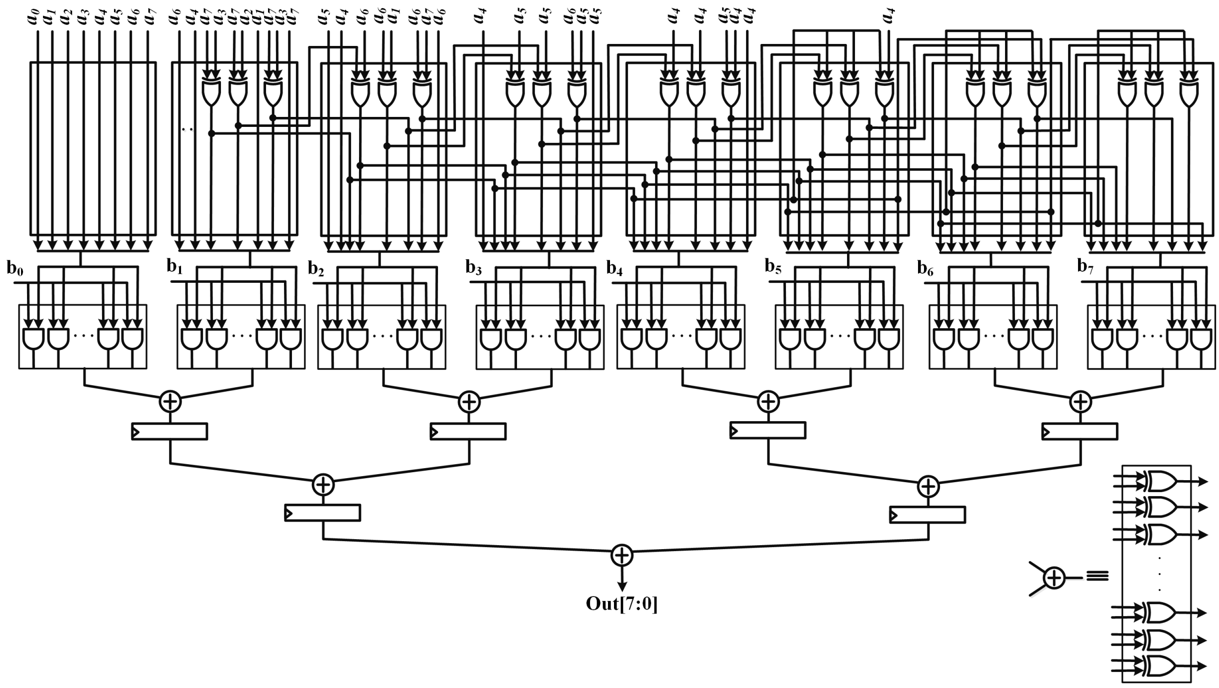

Figure 5. Proposed pipelined bit-parallel multiplier for GF(2^8) by $P(x) = x^8 + x^4 + x^3 + x + 1$.

$$\begin{aligned} C(x) = A(x)B(x) &= A(x)(B_{odd}(x) + B_{even}(x)) \\ &= B_{odd}(x)A(x) + B_{even}(x)A(x) \\ &= C_{odd}(x) + C_{even}(x), \end{aligned}$$

where we let $C_{odd}(x) = B_{odd}(x)A(x)$ and $C_{even}(x) = B_{even}(x)A(x)$. For the case of odd number m, we have

$$\begin{aligned} C_{even}(x) &= b_{m-1}x^{m-1}A(x) + b_{m-3}x^{m-3}A(x)\, b_{m-5}x^{m-5}A(x) \\ &+ \cdots + b_2x^2A(x) + b_0A(x) \\ C_{odd}(x) &= b_{m-2}x^{m-2}A(x) + b_{m-4}x^{m-4}A(x)\, b_{m-6}x^{m-6}A(x) \\ &+ \cdots + b_3x^3A(x) + b_1xA(x) \\ &= x(b_{m-2}x^{m-3}A(x) + b_{m-4}x^{m-5}A(x)\, b_{m-6}x^{m-7}A(x) \\ &+ \cdots + b_3x^2A(x) + b_1A(x)) \end{aligned}$$

and for even m we have

$$\begin{aligned} C_{even}(x) &= b_{m-2}x^{m-2}A(x) + b_{m-4}x^{m-4}A(x)\, b_{m-6}x^{m-6}A(x) \\ &+ \cdots + b_2x^2A(x) + b_0A(x) \\ C_{odd}(x) &= b_{m-1}x^{m-1}A(x) + b_{m-3}x^{m-3}A(x)\, b_{m-5}x^{m-5}A(x) \\ &+ \cdots + b_3x^3A(x) + b_1xA(x) \\ &= x(b_{m-1}x^{m-2}A(x) + b_{m-3}x^{m4}A(x)\, b_{m-5}x^{m-6}A(x) \\ &+ \cdots + b_3x^2A(x) + b_1A(x)) \end{aligned}$$

So, we may write $C_{odd}(x) = x(C'_{even}(x))$. Then

$$C(x) = C_{even}(x) + C_{odd}(x) = C_{even}(x) + x(C'_{even}(x))$$

Therefore $C(x)$ is divided in to two parts $C_{even}(x)$ and $C_{odd}(x)$. Moreover $C_{odd}(x)$ is computed using a multiplication by x and $C'_{even}(x)$. So, the main block of the computation of $C(x)\ mod\ P(x)$ is performing

Table 2. 8-bit representation $x^i A(x)$

i	$x^i A(x)$
1	$(a_6, a_5, a_4, a_3 \oplus a_7, a_2 \oplus a_7, a_1, a_0 \oplus a_7, a_7)$
2	$(a_5, a_4, a_3 \oplus a_7, (a_2 \oplus a_7) \oplus a_6, a_1 \oplus a_6, a_0 \oplus a_7, a_7 \oplus a_6, a_6)$
3	$(a_4, a_3 \oplus a_7, (a_2 \oplus a_7) \oplus a_6, (a_1 \oplus a_6) \oplus a_5, (a_0 \oplus a_7) \oplus a_5, a_7 \oplus a_6, a_6 \oplus a_5, a_5)$
4	$(a_3 \oplus a_7, (a_2 \oplus a_7) \oplus a_6, (a_1 \oplus a_6) \oplus a_5, ((a_0 \oplus a_7) \oplus a_5) \oplus a_4, (a_7 \oplus a_6) \oplus a_4, a_6 \oplus a_5, a_5 \oplus a_4, a_4)$
5	$((a_2 \oplus a_7) \oplus a_6, (a_1 \oplus a_6) \oplus a_5, ((a_0 \oplus a_7) \oplus a_5) \oplus a_4, ((a_7 \oplus a_6) \oplus a_4) \oplus k_1, (a_6 \oplus a_5) \oplus k_1, a_5 \oplus a_4, a_4 \oplus k_1, k_1)$
6	$((a_1 \oplus a_6) \oplus a_5, ((a_0 \oplus a_7) \oplus a_5) \oplus a_4, ((a_7 \oplus a_6) \oplus a_4) \oplus k_1, ((a_6 \oplus a_5) \oplus k_1) \oplus k_2, (a_5 \oplus a_4) \oplus k_2, a_4 \oplus k_1, k_1 \oplus k_2, k_2)$
7	$(((a_0 \oplus a_7) \oplus a_5) \oplus a_4, ((a_7 \oplus a_6) \oplus a_4) \oplus k_1, ((a_6 \oplus a_5) \oplus k_1) \oplus k_2, ((a_5 \oplus a_4) \oplus k_2) \oplus k_3, (a_4 \oplus k_1) \oplus k_3, k_1 \oplus k_2, k_2 \oplus k_3, k_3)$
8	$[a_1 \oplus a_8, a_0 \oplus a_7, a_9 \oplus a_6, a_8 \oplus a_5, a_7 \oplus a_4, a_6 \oplus a_3, a_5 \oplus (a_2 \oplus a_9), a_4, a_3, a_2 \oplus a_9]$
9	$[a_0 \oplus a_7, a_9 \oplus a_6, a_8 \oplus a_5, a_7 \oplus a_4, a_6 \oplus a_3, a_5 \oplus (a_2 \oplus a_9), a_4 \oplus (a_2 \oplus a_9), a_3, a_2 \oplus a_9, a_1 \oplus a_8]$

$k_1 = a_3 \oplus a_7$, $k_2 = (a_2 \oplus a_7) \oplus a_6$, $k_3 = (a_1 \oplus a_6) \oplus a_5$

$x^{2i}A(x)$, for $i = 1, 2, \ldots, m/2$. This method decreases the number of multiplications by powers of x to about half compared to that of the first method. The proposed structures of multiplication in binary finite fields by independent computation of the even powers of x are shown in Figure 6.a and Figure 6.b. As seen in Figure 6 after computing $C_{even}(x)$ and $C'_{even}(x)$, C_{odd} is computed by multiplication of $C'_{even}(x)$ by x. Also, $C_{even}(x)$ and $x(C'_{even}(x))$ are added to each other to perform the output. For example, Figure 7 shows the proposed structure of multiplication over GF(2^{10}) by the irreducible trinomial $P(x) = x^{10} + x^3 + 1$. The important part of the critical path in the multiplier structure is in the XOR tree part. Main focus is to shorten this part of the path with fewer number of the registers. Therefore, the pipeline registers are used to shorten critical path delay in the XOR tree of the proposed structure. For example, Figure 8 shows a pipelined XOR tree with length 8. As known for GF(2^m), the critical path of XOR trees for methods 1 and 2 are $(log_2^m)T_X$ and $(log_2^{m/2})T_X$, respectively. In both methods, the XOR tree structure is 3 stages pipelined by two row registers. Therefore, the tree paths in methods 1 and 2 of lengths $(log_2^m)T_X$ and $(log_2^{m/2})T_X$ are converted to three shorter paths of lengths $(\left(\left\lceil \frac{log_2^m}{2} \right\rceil - 1\right) T_X, \left\lceil \frac{log_2^m}{2} \right\rceil T_X$ and $T_X)$ and $(\left(\left\lceil \frac{log_2^{\frac{m}{2}}}{2} \right\rceil - 1\right) T_X, \left\lceil \frac{log_2^{\frac{m}{2}}}{2} \right\rceil T_X$ and $2T_X)$, respectively. According to the structures of multipliers, the delay of the first part of pipelined XOR trees are added to delays of the AND-network and computational part of the multiplication by powers of x. In the case of using irreducible pentanomial, the critical path delays of the proposed architectures after pipelining for methods 1 and 2 are $T_A + (4 + (\left\lceil \frac{log_2^m}{2} \right\rceil - 1))T_X$ and $T_A + (4 + (\left\lceil \frac{log_2^{\frac{m}{2}}}{2} \right\rceil - 1))T_X$, respectively. In addition, for irreducible trinomial, the critical path delays are $T_A + (2 + (\left\lceil \frac{log_2^m}{2} \right\rceil - 1))T_X$ and $T_A + (2 + (\left\lceil \frac{log_2^{\frac{m}{2}}}{2} \right\rceil - 1))T_X$. The proposed architectures have an initial 2 clock cycle latency, and after latency produce each result in every clock cycle. The circuits have advantages of very low latency and very high throughput.

3 Comparison and Result Analyses

In this section, we give a comprehensive comparison between this work and other FPGA-based designs, up to our best knowledge. The comparison approach is based on hardware resources, critical path delay, latency (based on the number of clock cycle), throughput, hardware utilization and power consumption of FPGA implementation. The work has been successfully verified, synthesized, using Xilinx ISE 11 by Virtex-4 XC4VLX200, XC4VLX40, and Virtex-2 XC2V6000 FPGAs. In Table 3, the values of hardware utilization including the number of 2-input XOR gate, 2-input AND gate, and D flip-flop are provided for several related works. In addition, in this table the structure of multipliers and the generating polynomials are given. For both methods, the critical path delay of the proposed pipelined structure over GF(2^{233}) constructed by trinomial $P(x) = x^{233} + x^{74} + 1$ is $T_A + 5T_X$. Moreover, for GF(2^{163}) by pentanomial $P(x) = x^{163} + x^7 + x^6 + x^3 + 1$, the critical path delay is $T_A + 7T_X$. It can be seen that critical path delays or time complexity of the proposed methods are the smallest compared with other existing bit-parallel designs. The results show that an overall improvement in speed is obtained. For the proposed design, when compared with the systolic designs in [25], [30] and [31] it requires nearly the same hardware in terms of the number of 2-input XOR and AND gates. However, the number of flip-flops in our design is $\frac{m}{8} + 2m$, compared to $(4m^2 + 3m)$, $3(m + 1)^2$ and $3m^2 + 2m - 2$ that are used in [25], [30] and [31], respectively. The latency of the proposed design is two clock cycles, while for the designs in [30] and [31] the latency is $m + 1$ clock cycles.

The proposed trinomial basis multiplier has nearly the same area as the trinomial basis bit-parallel designs presented in [6], [9], [17], [19], [21], [24], [29] and [31]. But, the critical path delay of proposed design is better, In these works the critical path is loga-

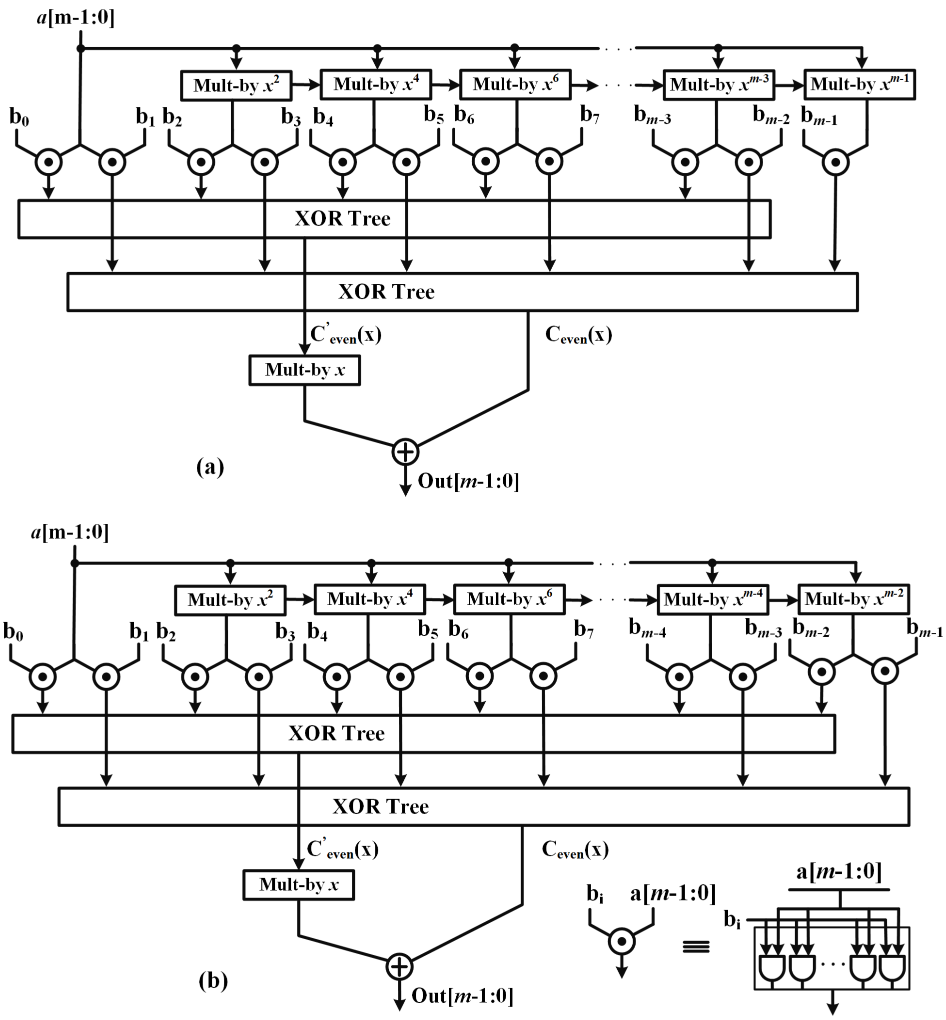

Figure 6. Proposed structures of bit-parallel multiplication over GF(2^m) using multiplication by even powers of x(a) $m$is odd, (b) $m$is even.

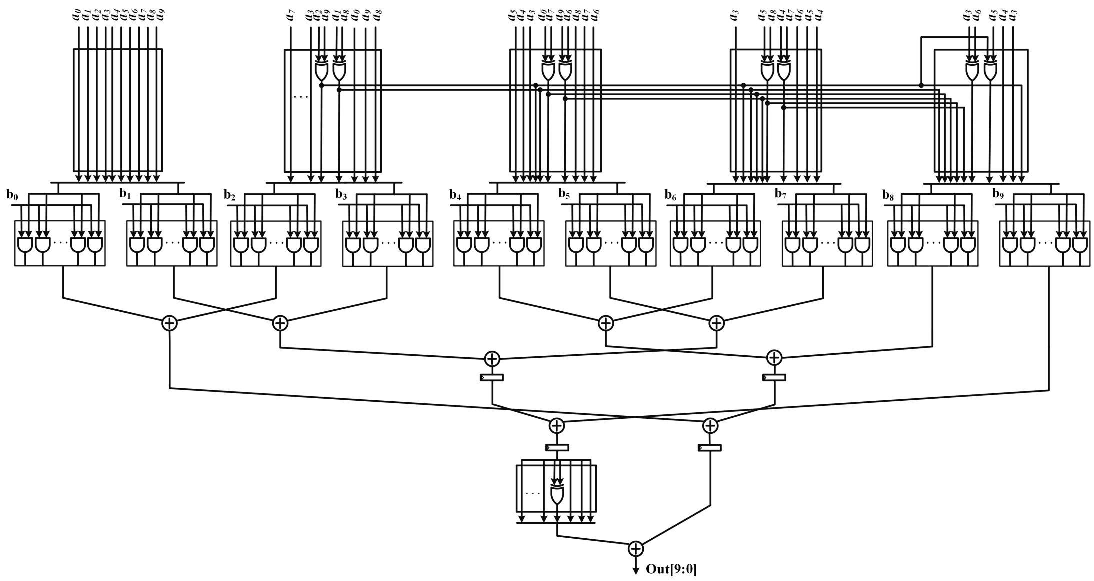

Figure 7. The second proposed structure of the pipelined bit-parallel multiplication over GF(2^{10}) by $P(x) = x^{10} + x^3 + 1$.

Table 3. Comparison of hardware resources

Works	# 2-input XOR	# 2-input AND	# FF	Structure	Generating Polynomial/Basis
[1]	$2m+k-2$	$2m-1$	$3m+k-1$	Bit-serial	Trinomial
[2]	$4m+4$(also$2m-2$ 3-input XOR)	$7m-4$	$6m$	Bit-parallel	General form
[3] $w=2$	m^2+2m	m^2	m^2	Digit-serial	Pentanomial
[4]	m^2-1	m^2	—	Bit-parallel	Trinomial
[6]	$\frac{3m^2}{4}+4m+k-\frac{23}{4}$	$\frac{3m^2+2m-1}{4}$	—	Bit-parallel	Trinomial
[9] Hybrid Karatsuba($m=233$)	47350	9435	—	Bit-parallel	Trinomial
[9] Masked Hybrid Karatsuba ($m=233$)	143937	28485	—	Bit-parallel	Trinomial
[10]	$(m+1)q$	$(m+1)q$	—	Bit-parallel	All One polynomial
[11]	$2mw$	$2mw$	$4m$	Digit-serial	Pentanomial
[13]	$m\sqrt{m}$	$\sqrt{dm}(2+m)+d$	$\sqrt{\frac{m}{d}}(2m+\ d-1)+2m$	Digit-serial	Trinomial
[15]	$2m$	$3m$	$4m+2$ (# MUX 2to $1=2m$)	Bit-serial	General form
[16]	$2m^2$	$2m^2$	$3m^2$	Bit-parallel	General form
[17]	$m(\frac{m}{2}-\frac{1}{2})-1$	0	# MUX 2to $1=m(\frac{m}{2})$	Bit-parallel	Trinomial
[18] (#Tri-state buffer=m and #inverter $=m$)	$2m$	$4m$and #OR $=m$	$3m$	Bit-serial	General form
[19]	m^2+m (3-input XOR)	$2m^2+3m$	$3m^2+4m$	Bit-parallel	Trinomial
[20]	$\frac{2m^2}{3}+\frac{22m}{3}$	$\frac{2m^2}{3}+\frac{8m}{3}-\frac{4}{3}$	—	Bit-parallel	General form
[21]	$\frac{16}{3}m^{log_3^6}-\frac{22}{3}m+2$	$m^{log_3^6}$	—	Bit-parallel	Trinomial
[24]	m^2-1	m^2	—	Bit-parallel	Trinomial
[24]	m^2+2m-3	—	—	Bit-parallel	Pentanomial
[25]	m^2+2m	m^2	$4m^2+3m$	Bit-parallel	General form
[27]	$\frac{25}{4}m^{log_3^6}-5m+1.5$	$m^{log_3^6}$	—	Bit-parallel	Trinomial
[29]	m^2-1	m^2	—	Bit-parallel	Trinomial
[30]	$(m+1)^2$	$(m+1)^2$	$3(m+1)^2$	Bit-parallel	All One polynomial
[31]	m^2+m-1	m^2	$3m^2+2m-2$	Bit-parallel	Trinomial
[44]	m^2+m-3	m^2	—	Bit-parallel	Type 1, 2 Polynomials
[44]	m^2+m-3	m^2	—	Bit-parallel	Type 3 Polynomials
[45]	$m+1$	$m+1$ (# MUX 2 to $1=m+3$)	$3m+\left\lceil log_2^m \right\rceil+3$	Bit-serial	All One polynomial
[36]	$\leq\frac{T+4}{4}m(m-1)$	m^2	$2m$	Bit-parallel	Gaussian normal basis
[37]	$\leq\frac{T+1}{2}m(m-1)$	m^2	$3m$	Bit-parallel	Gaussian normal basis
[38]	$\leq Tm(m-1)+m$	m^2	$3m$	Bit-parallel	Normal basis
[39]	$\leq\frac{T+1}{2}m(m-1)$	m^2	$2m$	Bit-parallel	Normal basis
[40]	$1.5m(m-1)$	m^2	$2m$	Bit-parallel	Normal basis
[41] type 1	$wm+m+w+1$	$wm+m+w+1$	$2m$	Digit-serial	Normal basis
[41] type 2	$w(m+\left\lfloor\frac{m}{2}\right\rfloor)$	$wm+m$	$2m$	Digit-serial	Normal basis
[41] type 2	$\leq\frac{m}{2}(3m-3)$	m^2	$2m$	Bit-parallel	Normal basis
[43]	$(m^2+3m-2)/2$	$m(m-1)/2$ (MUX 4 to 1)	—	Bit-parallel	Normal basis
Proposed method 1	m^2+m	m^2	$\frac{m}{8}+2m$	Bit-parallel	Trinomial
Proposed method 1	m^2+3m	m^2	$\frac{m}{8}+2m$	Bit-parallel	Pentanomial

Note: $q=K*\left\lfloor\frac{m+1}{K}\right\rfloor+\left\lfloor log_2^K\right\rfloor$, $K$is a small positive integer ($K\leq 10$); w, d: digit size; k is in irreducible trinomial $P(x)=x^m+x^k+1$.

rithmically dependent on the operands length. While, in the proposed work logarithm of operands length in the critical path delay formula is divided by 2. For example, in [27] and [29] the critical path delay is $T_A+(3\lceil log_2^m\rceil+1)T_X$ and $T_A+(2+\lceil log_2^{m-1}\rceil)T_X$, respectively whereas it is $T_A+(2+(\left\lceil\frac{log_2^m}{2}\right\rceil-1))T_X$ and $T_A+(2+(\left\lceil\frac{log_2^{\frac{m}{2}}}{2}\right\rceil-1))T_X$ for the proposed methods 1 and 2, respectively. The proposed pentanomial basis multiplier, has shorter critical path delay compared with the pentanomial basis bit-parallel design presented in [24]. The critical path delay in [24] is $T_A+(4+\lceil log_2^{m-1}\rceil)T_X$ compared to $T_A+(4+(\left\lceil\frac{log_2^m}{2}\right\rceil-1))T_X$ and $T_A+(4+(\left\lceil\frac{log_2^{\frac{m}{2}}}{2}\right\rceil-1))T_X$, respectively, for the proposed methods 1 and 2 in pentanomial basis case.

Total time delay of Non-redundant KOA multiplier in [7] is the same as parallel KOA multiplier, but with lower number of total gates. In addition, the efficiency of multiplier in [7] depends on the hamming-weight of degree m.

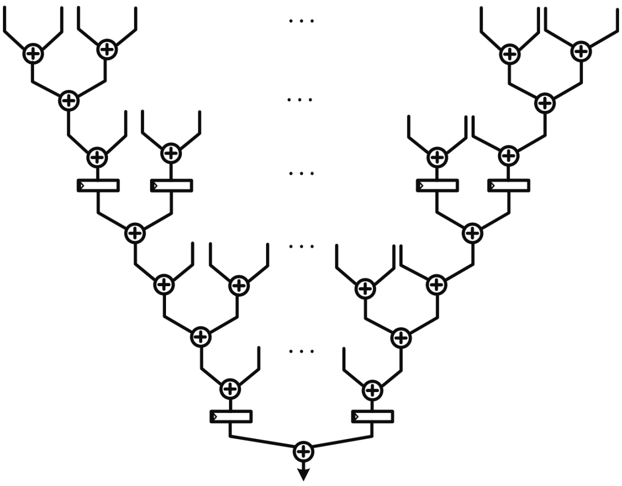

Figure 8. Pipelined XOR tree with length of 8.

As seen in the above table the values of the hardware utilization in the proposed structures are reasonable and nearly the same compared to the other polynomial bit-parallel designs.

In this work, FPGA implementations of the bit-parallel multipliers over finite fields $GF(2^{163})$ and $GF(2^{233})$ generated by the irreducible pentanomial $P(x) = x^{163} + x^7 + x^6 + x^3 + 1$ and trinomial $P(x) = x^{233} + x^{74} + 1$ are reported. These binary extension fields are recommended by National Institute of Standards and Technology (NIST) for elliptic curve cryptosystems. Comparison of the FPGA synthesis results are provided in Table 5.

Notice several works, for example [1–4], [6], [10], [13], [16, 17] and [20, 21] do not have hardware implementation. Moreover, some FPGA based related works did not completely reported the synthesis results. Table 5 shows the maximum operation frequencies of the proposed structures, which are the highest operation frequencies among all FPGA based reported works. The proposed architectures after an initial two clock cycles latency produce each result in every clock cycle. The throughput is computed by the following equation in which the number of processed bits is based on field size.

$$Throughput = \frac{\text{Number of processed bits}}{\text{Clock period} \times \text{Number of Cycles}} = \frac{\text{Number of processed bits}}{\text{Time}}$$

Since target devices are different in different works, to have a fair comparison with works that used older FPGAs, for example, Virtex-2 FPGA family, the proposed design is also implemented on Virtex-2 using Xilinx ISE 10.1 tool. Also, to compare the design with works presented in [33] and [26] which are implemented on Virtex-4 XC4VLX40 with speed grade -12, it is implemented on this FPGA family too.

Table 4. Comparison of the critical path delay and latency

Work	Critical Path Delay	Latency (Clock Cycle)
[1]	$T_A + (2 + log_2^m)T_X$	1
[2]	$((\frac{m}{2})+2)(2T_A + 2T_X + T_L)$	—
[3]	$4T_X$	$\frac{m}{4} + 2$
[4]	$T_A + (log_2^{m-1}+1)T_X$	—
[6]	$T_A + (3 + \lceil log_2^{m-1} \rceil)T_X$	—
[7] ($m = 163$)	$20Tx + T_A$	—
[10]	$h^* (T_A + T_X + T_L)$	h
[11]	$w(T_A + T_X)$	14
[12] Bit-serial	—	80
[13]	$T_A + (log_2^{(d+1)})T_X + T_L$	$2\sqrt{\frac{m}{d}}$
[15]	$T_M + 2T_X$	$0.664m$
[16]	$m(T_A + T_X + T_L)$	m
[17]	$T_M + (\lceil log_2^{\frac{m}{2}} + 1 \rceil)T_X$	—
[18]	$T_A + T_X + T_{Tri\_state}$	m
[20]	$T_A + (3 + \lceil log_2{}^{\frac{m-1}{3}} \rceil)T_X$	—
[21]	$T_A + (3 \lceil log_3^m \rceil)T_X$	—
[23] Digit-serial ($m = 163$)	—	144
[24] Trinomial	$T_A + (2 + \lceil log_2^{m-1} \rceil)T_X$	—
[24] Pentanomial	$T_A + (4 + \lceil log_2^{m-1} \rceil)T_X$	—
[27]	$T_A + (3 \lceil log_2^m \rceil + 1)T_X$	—
[29]	$T_A + (2 + \lceil log_2^{m-1} \rceil)T_X$	—
[30]	$(m+1)(T_A + T_X + T_L)$	—
[31]	$(m+1)(T_A + T_X + T_L)$	—
[36]	$T_A + (\lceil log_2^T \rceil) + \lceil log_2^m \rceil)T_X$	1
[37]	$T_A + (\lceil log_2^T \rceil) + \lceil log_2^m \rceil)T_X$	1
[38]	$T_A + (\lceil log_2^T \rceil) + \lceil log_2^m \rceil)T_X$	1
[39]	$T_A + (\lceil log_2^{Tm-T+1} \rceil)T_X$	1
[40]	$T_A + (1+\lceil log_2^m \rceil)T_X$	1
[41] type 1	$\leq 2T_A + (3+\lceil log_2^{w-1} \rceil)T_X$	w
[41] type 2	$\leq 2T_A + (3+\lceil log_2^{w-1} \rceil)T_X$	w
[42] type 2	$T_A + (2 + \lceil log_2^{2m-1} \rceil)T_X$	1
[43]	$T_{M4} + (2 + \lceil log_2^{m-1} \rceil)T_X$	1
[44] Type 1, 2 Polynomials	$T_A + (3+\lceil log_2^{m-3} \rceil)T_X$	1
[44] Type 3 Polynomials	$T_A + (3+\lceil log_2^{m-1} \rceil)T_X$	1
[45]	$Tx + T_A$	$m+\lceil log_2^m \rceil+1$
Proposed method 1 Trinomial	$T_A + (2 + (\lceil \frac{log_2^m}{2} \rceil - 1))T_X$	2
Proposed method 1 Pentanomial	$T_A + (4 +(\lceil \frac{log_2^m}{2} \rceil - 1))T_X$	2
Proposed method 2 Trinomial	$T_A + (2 + (\lceil \frac{log_2^{\frac{m}{2}}}{2} \rceil - 1))T_X$	2
Proposed method 2 Pentanomial	$T_A + (4 +(\lceil \frac{log_2^{\frac{m}{2}}}{2} \rceil - 1))T_X$	2

Note: T_A, T_X, T_L, T_M, T_{M4}, T_{Tri_state} denote the Time delay of an AND gate, XOR gate, one bit Latch, multiplexer 2 to 1, multiplexer 4 to 1 and Tri-state buffer respectively; d, w: digit size; $h=[(m + 1)/k] + [log_2^k]$.

In Table 5 works [8], [12], [15], [18], and [45] have bit-serial structure with low hardware resources and low critical path delay compared to other digit-serial and bit-parallel structures. However, due to higher number

Table 5. Comparison of the critical path delay and latency

Work	FPGA	Area	Time(ns)	Freq.(Mhz)	Throughput(Mbps)
[5] GF(2^{163})	Virtex-5 —	LUTs 7786	5.468	—	29813
[8] GF(2^{239}) Bit-serial	Virtex-E XCV300	Slices 359	3100	75	75
[12] GF(2^{160}) Bit-serial	Virtex-E XCV800	Slices 1049	—	—	—
[14] GF(2^{191})	Virtex-E XCV3200E	Slices 8721	—	—	—
[35] GF(2^{193})	Virtex-E XCV3200E	Slices 8753	43.1	—	4478
[22] GF(2^{191})	Virtex-E XCV2600	Slices 8721	45889	—	4
[23] GF(2^{160}) Digit-serial	Virtex-E XCV2000E	Gates 21600 (LUTs 3344, FF 64)	—	—	—
[5] GF(2^{239})	Virtex-2 —	Slices 10510	10.710	—	22316
[11] GF(2^{163}) Digit-serial	Virtex-2 XC2VP30	LUTs 12210	197.18	71	827
[15] GF(2^{233}) Bit-serial	Virtex-2 XC2V1000	CLBs 523	787	197	296
[15] GF(2^{163}) Bit-serial	Virtex-2 XC2V1000	CLBs 330	543	199	297
[18] GF(2^{160}) Bit-serial	Virtex-2 XC2V500	CLBs 635 (FF 481)	731	219	219
[45] GF(2^{160}) Bit-serial	Virtex-2 XC2V500	CLBs 296 (489 FF)	740	227	216
[28] GF(2^{233}) Digit-serial (d=1)	Virtex-2 XC2V6000	Slices 246 (LUTs 484, FF 477)	992.58	234.8	235
[28] GF(2^{233}) Digit-serial (d=32)	Virtex-2 XC2V6000	Slices 4457 (LUTs 7110, FF 1349)	52.72	151.6	4415
[19] GF(2^{233}) Hybrid Karatsuba	Virtex-2 XC2V6000	LUTs 11746, FF 13941	11.07	90.33	21047
[19] GF(2^{233}) Classical	Virtex-2 XC2V6000	LUTs 37296, FF 37552	13	77	17941
[34] GF(2^{163})	Virtex-2 XC2V6000	Slices 12640 (LUTs 21058)	15.496	—	10519
[9] GF(2^{233})	Virtex-4 —	Slices 30435	17	—	13706
[36] GF(2^{163}) DL-SIPO Digit-serial (d=55)	Virtex-4 XC4VLX100	Slices 9323 (LUTs 16715, FF 326)	20.1	149	8096
[36] GF(2^{163}) DL-PISO Digit-serial (d=55)	Virtex-4 XC4VLX100	Slices 9678 (LUTs 17348, FF 419)	21.9	137	7444
[33] GF(2^{163})	Virtex-4 XC4VLX40	Slices 8665 (LUTs 15356)	8.284	120.7	19674
[26] GF(2^{163})	Virtex-4 XC4VLX40	Slices 4651 (LUTs 8624)	8.487	117.83	19206
[1]Proposed method 1 GF(2^{163})	Virtex-2 XC2V6000	Slices 11958 (LUTs 19916, FF 3749)	8.832	226.467	18457
[1]Proposed method 1 GF(2^{233})	Virtex-2 XC2V6000	Slices 25555 (LUTs 48712, FF 7456)	7.758	257.819	30035
[1]Proposed method 1 GF(2^{163})	Virtex-4 XC4VLX40	Slices 10515 (LUTs 18250, FF 3749)	6.816	290.394	23667
[1]Proposed method 1 GF(2^{163})	Virtex-4 XC4VLX200	Slices 10515 (LUTs 18250, FF 3749)	7.814	255.951	20860
[1]Proposed method 1 GF(2^{233})	Virtex-4 XC4VLX200	Slices 21195 (LUTs 36812, FF 7456)	7.191	278.133	32402

[1]. The results for proposed method 1 are equal to the proposed method 2.

of clock cycles throughput of bit-serial structure is less than those of digit-serial and bit-parallel structures. Digit-serial architectures in [23], [11], [28], and [36] have better timing performance compared to bit-serial structure, but their hardware consumption in terms of Slices (LUTs and FF) is higher. The bit-parallel architectures proposed in [5], [9], [19], [26], [33] and [34] have lowest number of clock cycles, but they include a large amount of XOR and AND logical gates and flip-flops.

In terms of hardware utilization, considering the values of other bit-parallel structures, the proposed bit-parallel structures have reasonable values. Also, compared to the works presented in [33] and [26] the proposed structures have better performance in terms of frequency, execution time and throughput, but it uses more slices. Similarly, for the target device of Virtex-2 FPGA, the results of proposed method are suitable. For example in terms of execution time, the proposed method shows 44.962ns and 3.312ns time reduction, respectively, when compared with those of [28] and [19] for GF(2^{233}), and also 6.664ns time reduction when compared to that of [34] for GF(2^{163}). On a Virtex 4 FPGA, the masked multiplier with the Hybrid Karatsuba implementation for GF(2^{233}) requires 30435 slices [9], while the proposed Multiplier needs 21195 slices. Moreover, the delay of [9] is two times more than that of our design. Also throughput in [9] is 13706 Mbps, compared to 32402 Mbps in present work.

Table 6 shows the power consumption of the present work based on method 1 by Virtex-4 XC4VLX200, and also of the circuit presented in [18] by “CoolRunner XPLA3” CPLD. The power consumption is measured by the Xilinx Power Estimator (XPE) tool at different operational frequency.

The graphical representations of the power by function (Typical, 1.2V, 26°C), Power vs. voltage (Typical, 26°C), and Power vs. Temperature (Typical, 1.2V), for the proposed structure for GF(2^{163}) and GF(2^{233})

Table 6. Comparison of the power consumption.

Work	This work Method 1 $GF(2^{163})$				This work Method 1 $GF(2^{233})$				[18] $GF(2^{160})$
Frequency(MHz)	50	100	150	200	50	100	150	200	214
Quiescent(static power)(w)	0.992	0.994	0.996	0.999	0.994	0.997	1.001	1.005	—
Dynamic(w)	0.188	0.303	0.418	0.533	0.286	0.468	0.649	0.830	—
Total(w)	1.18	1.297	1.414	1.532	1.28	1.465	1.65	1.835	3.523

are shown in Figure 9 (a)-(c) and Figure 9 (d)-(f) respectively. The power consumption for both methods 1 and 2 are equal.

4 Conclusions

In this paper two fast and pipelined architectures for bit-parallel polynomial basis multipliers in binary finite fields are presented. The implementations are performed over finite fields constructed by irreducible trinomials and pentanomials. The structures of the multipliers are based on a parallel and independent computation of powers of x as the polynomial variable. To speed up the multiplication operation, two inputs of the multiplier are simultaneously applied in parallel form. In addition, the pipelining technique is used to shorten critical path and to increase speed and performance. In the proposed methods after an initial 2 clock cycle latency, each result is produced in every clock cycle, so the circuits provide very low latency and high throughput.

Acknowledgment

The author would like to thank the reviewers for their comments. This research was in part supported by a grant from IPM (No. 93050416).

References

[1] Arash Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases", Cryptographic Hardware and Embedded Systems-CHES 2008, Vol. 5154, pp. 300-314.

[2] Che-Wun Chiou and Huey-Lin Jeng, "Parallel Algorithm for Polynomial Basis Multiplier in $GF(2^m)$ Fields", Tamkang Journal of Science and Engineering, Vol. 11, No. 2, 2008, pp. 211-218.

[3] XIE Jia-feng, HE Jian-jun, GUI Wei-hua, "Low latency systolic multipliers for finite field $GF(2^m)$ based on irreducible polynomials", Journal of Central South University, Vol. 19, Iss. 5, 2012, pp. 1283-1289.

[4] Huapeng Wu, "Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis", IEEE Transactions on Computers, Vol. 51, No. 7, July 2002, pp. 750-758.

[5] Eduardo Cuevas-Farfan, Miguel Morales-Sandoval, Alicia Morales-Reyes, Claudia Feregrino-Uribe, Ignacio Algredo-Badillo, Paris Kitsos, René Cumplido, "Karatsuba-Ofman Multiplier with Integrated Modular Reduction for $GF(2^m)$", Advances in Electrical and Computer Engineering, Vol. 13, No. 2, 2013, pp. 3-10.

[6] M. Elia, M. Leone and C.Visentin, "Low complexity bit-parallel multipliers for $GF(2^m)$ with generator polynomial $P(x) = x^m + x^k + 1$", Electronics Leters 1st April 1999 Vol. 35 No. 7, pp. 551-552.

[7] Nam Su Chang, Chang Han Kim, Young-Ho Park , and Jongin Lim, "A Non-redundant and Efficient Architecture for Karatsuba-Ofman Algorithm", Proceedings of the 8th International Conference on Information Security (ISC), Singapore, September 20-23, Springer-Verlag Berlin Heidelberg, Vol. 3650, 2005, pp. 288-299.

[8] Mario Alberto García-Martínez, Rubén Posada-Gómez, Guillermo Morales-Luna and Francisco Rodríguez-Henríquez, "FPGA Implementation of an Efficient Multiplier over Finite Fields $GF(2^m)$", Proceedings of the IEEE International Conference on Reconfigurable Computing and FPGAs, 2005, pp.21-26.

[9] Chester Rebeiro and Debdeep Mukhopadhyay, "Hybrid Masked Karatsuba Multiplier for $GF(2^{233})$", 11th IEEE VLSI Design and Test Symposium, Kolkata, August 2007.

[10] Che Wun Chiou and Liuh Chii Lin, "Fast Array Multiplications over $GF(2^m)$ Fields with Multiple Speeds", Tamkang Journal of Science and Engineering, Vol. 7, No 3, 2004 , pp. 139-144.

[11] Junfeng Fan and Ingrid Verbauwhede, "A Digit-Serial Architecture for Inversion and Multiplication in $GF(2^m)$", IEEE Workshop on Signal Processing Systems, 8-10 Oct. 2008, pp. 7-12.

[12] Lejla Batina, Nele Mentens, Sıddıka Berna Ors, Bart Preneel, "Serial Multiplier Architectures over $GF(2^m)$ for Elliptic Curve Cryptosystems", 12th IEEE Electrotechnical Conference, Vol.2 2004, pp. 779-782.

[13] Jeng-Shyang Pan, Chiou-Yng Lee and Pramod Kumar Meher, "Low-Latency Digit-Serial and Digit-Parallel Systolic Multipliers for Large Binary Extension Fields",IEEE Transactions on Circuits and Systems I: Regular Papers, Dec. 2013, pp. 3195-3204.

[14] Nazar A. Saqib, Francisco Rodriguez-Henriquez

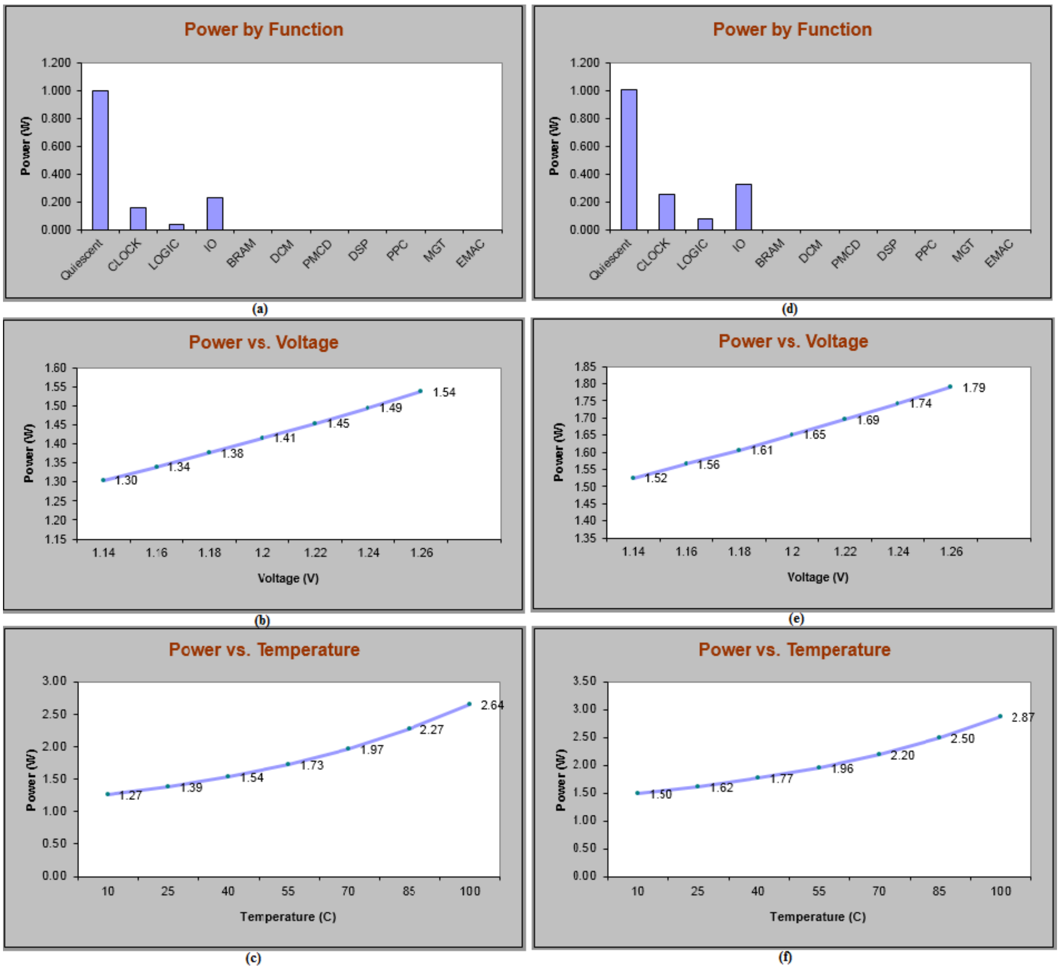

Figure 9. The graphical representations of the (a): power by function(Typical, 1.2V, 26°C), (b):On-Chip Power vs. voltage (Typical, 26°C) , and (c): Power vs. Temperature(Typical, 1.2V) for GF(2^{163}); and also similarly (d)-(f) for GF(2^{233}) in the proposed structure.

and Arturo Diaz-Perez, "A Parallel Architecture for Fast Computation of Elliptic Curve Scalar Multiplication over GF(2^m)" 18th International Parallel and Distributed Processing Symposium, 26-30 April 2004.

[15] George N. Selimis, Apostolos P. Fournaris, Harris E. Michail, Odysseas Koufopavlou, "Improved throughput bit-serial multiplier for GF(2^m) fields", Integration, the VLSI Journal 42, 2009, pp. 217-226.

[16] Che-Wun Chiou, Chiou-Yng Lee and Jim-Min Lin, "Finite Field Polynomial Multiplier with Linear Feedback Shift Register", Tamkang Journal of Science and Engineering, Vol. 10, No. 3, 2007, pp. 253-264.

[17] Chiou-Yng Lee, Che Wun Chiou , Jim-Min Lin, "Low-complexity bit-parallel dual basis multipliers using the modified Booths algorithm", Computers and Electrical Engineering Vol. 31, 2005, pp. 444-459.

[18] Ali Zakerolhosseini, Morteza Nikooghadam, "Low-power and high-speed design of a versatile bit-serial multiplier in finite fields GF(2^m)", Integration, the VLSI Journal Vol. 46, 2013, pp. 211-217.

[19] C. Grabbe, M. Bednara, J. Teich, J. von zur Gathen, J. Shokrollahi "FPGA Designs of Parallel High Performance GF(2^{233}) Multipliers" International Symposium on Circuits and Systems, 2003, Vol. 2, pp. 268-271.

[20] Yin Li, Gongliang Chen, Xiao-ning Xie: "Low complexity bit-parallel GF(2^m) multiplier for all-one polynomials", IACR Cryptology ePrint Archive 2012: 414 (2012).

[21] Haining Fan, Jiaguang Sun, Ming Gu and Kwok-Yan Lam, "Overlap-free Karatsuba-Ofman Polynomial Multiplication Algorithms", IET Information security, Vol. 4, No. 1, 2010, pp. 8-14.

[22] Sameh M. Shohdy, Ashraf B. El-Sisi, and Nabil Ismail, "Hardware Implementation of Efficient Modified Karatsuba Multiplier Used in Elliptic Curves", International Journal of Network Security, Vol. 11, No. 3, Nov. 2010, pp.155-162.

[23] Mohammed Benaissa and Wei Ming Lim, "Design

of Flexible GF(2^m) Elliptic Curve Cryptography Processors", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, No. 6, June 2006, pp. 659-662.

[24] Arash Reyhani-Masoleh, and M. Anwar Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF(2^m)", IEEE Transactions on Computers, Vol. 53, No. 8, August 2004, pp. 945-959.

[25] Chiou-Yng Lee, Chin-Chin Chen,Yuan-Ho Chen and Erl-Huei Lu, "Low-Complexity Bit-Parallel Systolic Multipliers over GF(2^m)", IEEE International Conference on Systems, Man, and Cybernetics, 2006, pp. 1-6.

[26] Gang Zhou, Harald Michalik, and László Hinsenkamp, "Complexity Analysis and Efficient Implementations of Bit Parallel Finite Field Multipliers Based on Karatsuba-Ofman Algorithm on FPGAs", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 18, No. 7, July 2010, pp. 1057-1066.

[27] Haining Fan and M. Anwar Hasan, "A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields", IEEE Transactions on Computers, Vol. 56, No. 2, February 2007, pp. 224-233.

[28] Miguel Morales-Sandoval, Claudia Feregrino-Uribe, René Cumplido, Ignacio Algredo-Badillo, "An area/performance trade-off analysis of a GF(2^m) multiplier architecture for elliptic curve cryptography", Computers and Electrical Engineering, Vol. 35, 2009, pp. 54-58.

[29] Huapeng Wu, "Bit-Parallel Finite Field Multiplier and Square Using Polynomial Basis", IEEE Transactions Computers, Vol. 51, 2002, pp. 750-758.

[30] Lee, C. Y., Lu, E. H. and Lee, J. Y., "Bit-Parallel Systolic Multipliers for GF(2^m) Fields Defined by All-One and Equally-Spaced Polynomials," IEEE Transactions Computers, Vol. 50, 2001, pp. 385-393.

[31] Lee, C. Y., "Low Complexity Bit-Parallel Systolic Multiplier Over GF(2^m) Using Irreducible Trinomials," IEE Proc. Comput. Digit. Tech., Vol. 150, 2003, pp. 39-42.

[32] Bahram Rashidi, Reza Rezaeian Farashahi, Sayed Masoud Sayedi, "High-speed and Pipelined Finite Field Bit-Parallel Multiplier over GF(2^m) for Elliptic Curve Cryptosystems", Proceedings of the 11th International ISC Conference on Information Security and Cryptology (ISCISC), 3-4 Sept. 2014, pp. 15-20.

[33] G. Zhou, L. Li, and H. Michalik, "Area optimization of bit parallel finite field multipliers with fast carry logic on FPGAs", Proceedings of the International Conference on Field Program. Logic and Applications (ICFPL), Sep. 2008, pp. 671-674.

[34] W. N. Chelton and M. Benaissa, "Fast elliptic curve cryptography on FPGA," IEEE Transactions Very Large Scale Integration (VLSI) System, Vol. 16, No.2, Feb. 2008, pp. 198–205.

[35] F. Rodríguez-Henríquez, N. A. Saqib, and N. Cruz-Cortés, "A fast implementation of multiplicative inversion over GF(2^m)", in Proceedings of the International Conference on Inf. Technol.: Coding Computer, 2005, pp. 574–579.

[36] Reza Azarderakhsh, Arash Reyhani-Masoleh, "Low-Complexity Multiplier Architectures for Single and Hybrid-Double Multiplications in Gaussian Normal Bases", IEEE Transactions on Computers, Vol. 62, No. 4, April 2013, pp. 744-757.

[37] Arash Reyhani-Masoleh, "Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases," IEEE Transactions Computers, Vol. 55, No. 1, Jan. 2006, pp. 34-47.

[38] A.H. Namin, H. Wu, and M. Ahmadi, "A Word-Level Finite Field Multiplier Using Normal Basis," IEEE Transactions Computers, Vol. 60, No. 6, June 2010, pp. 890-895.

[39] Arash Reyhani-Masoleh and M.A. Hasan, "A New Construction of Massey-Omura Parallel Multiplier over GF(2^m)" IEEE Transactions Computers, Vol. 51, No. 5, May 2002, pp. 511-520.

[40] C. K. Koç and B. Sunar, "An Efficient Optimal Normal Basis Type II Multiplier over GF(2^m)" IEEE Transactions Computers, Vol. 50, No. 1, Jan. 2001, pp. 83-87.

[41] Arash Reyhani-Masoleh, M. Anwar Hasan, "Low Complexity Word-Level Sequential Normal Basis Multipliers", IEEE Transactions on Computers, Vol. 54, No. 2, Feb. 2005, pp.98-110.

[42] Arash Reyhani-Masoleh, M. Anwar Hasan, "Efficient Digit-Serial Normal Basis Multipliers over Binary Extension Fields", ACM Transactions on Embedded Computing Systems, Vol. 3, No. 3, August 2004, pp. 575-592.

[43] Jenn-Shyong Horng, I-Chang Jou, Chiou-Yng Lee, "Low-complexity multiplexer-based normal basis multiplier over GF(2^m)", Journal of Zhejiang University Science A, 2009 Vol. 10, No.6, pp. 834-842.

[44] Huapeng Wu, "Bit-Parallel Polynomial Basis Multiplier for New Classes of Finite Fields", IEEE Transactions on Computers, Vol. 57, No. 8, August 2008, pp. 1023-1031.

[45] M. Nikooghadam, A. Zakerolhosseini, "Utilization of Pipeline Technique in AOP Based Multipliers with Parallel Inputs", Journal of Signal Processing Systems, Vol. 72, No. 1, pp. 57-62.

Bahram Rashidi was born in Boroujerd, Iran, in 1986. He received his B.S. degree in electrical engineering from Lorestan University, Iran, in 2009 and he received his M.S. from Tabriz University, Iran in 2011 also he is now Ph.D. student in Isfahan University of Technology (IUT). His research interests include hardware implementation for arithmetic of finite fields, cryptographic hardware, and VLSI circuits for elliptic curve cryptosystems.

Reza Rezaeian Farashahi is an assistant professor at department of Mathematical sciences at Isfahan University of Technology. He was a research fellow at the center for Advanced Computing Algorithms and Cryptography (ACAC) at Macquarie University. He obtained his Ph.D at the Eindhoven University of Technology, in Netherlands. Currently, his main research interests are algebraic aspects and computational number theory in cryptography and especially in elliptic and hyperelliptic curve cryptography. Specific areas he has worked on include: efficient arithmetic and implementation on elliptic curves over finite fields, computational problems on families of elliptic curves, number extractors for curves and Jacobians and pseudorandom bit generators.

Sayed Masoud Sayedi was born in Maragheh, Iran, in 1960. He received the B.Sc. and M.Sc. degrees in electrical engineering from Isfahan University of Technology (IUT), and the Ph.D. degree in electronics from Concordia University in 1986, 1988, and 1996, respectively. From 1988 to 1992, and then since 1997, he has been with IUT, where he is currently an associate professor in the Department of Electrical and Computer Engineering. His areas of interest include VLSI fabrication processes, low power VLSI circuits, and data converters.