Egyptian Informatics Journal 23 (2022) 105–112

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Low complexity design of bit parallel polynomial basis systolic multiplier using irreducible polynomials

Sakshi Devi*, Rita Mahajan, Deepak Bagai

Department of Electronics and Communication Engineering, Punjab Engineering College (Deemed to Be University), Chandigarh-160012, India

A R T I C L E I N F O

Article history:
Received 31 July 2020
Revised 29 May 2021
Accepted 9 July 2021
Available online 23 July 2021

Keywords:
Finite field
Cryptography
Polynomial basis
Systolic architectures
Nodes

A B S T R A C T

Encryption schemes like AES require finite field modular multiplication. The encryption speed is highly dependent on the performance of the finite field multiplier. Several high-speed systolic bit parallel multipliers with low area complexity have been proposed in the literature. In this paper, a modular multiplication algorithm is used to propose a bit parallel polynomial basis systolic multiplier which has achieved 89% less and 17% less area-delay product than the best existing multipliers. It has been observed that the area complexity of the proposed systolic multiplier for irreducible polynomials matches with the best-reported multiplier with 17% less time complexity. The results are further verified with the help of the FPGA implementation of the proposed multiplier for m = 8,163. Being generic, the proposed multiplier can be optimized further for trinomials and pentanomials.

1. Introduction

Galois field, also known as the finite field is named after famous French mathematician Evariste Galois. It forms the basis of various applications like cryptography, digital signal processing, pseudo-random number generation, switching theory, error-correcting codes, etc. Data can be effectively managed and manipulated if stored in Galois fields/finite fields. Finite fields are based on modular arithmetic. Modular arithmetic is extensively used in Mathematics and forms the basic building block of number theory. It is used in numerous practical applications like in calculating checksums and detecting errors for international standard book numbers (ISBNs), in modern commerce which employs public-key cryptography systems and also in error detection systems for bank identifiers.

Arithmetic operations like division, multiplication, addition, subtraction, and exponentiation can be computed on finite fields. Addition and subtraction are simply XOR operations. Division and exponentiation are performed using repeated multiply and square algorithms. However, multiplication is a very complex process and takes more time to do the computation. To perform the faster multiplication in finite fields, efficient multiplier architectures have to be designed [1]. There are three types of representation of finite field elements: Dual basis, Polynomial basis or canonical basis or standard basis and Normal Basis. Among all these representations, the polynomial basis is preferred because it can be matched to any system without any external circuitry. Moreover, it has a very low complexity than normal and dual basis multiplier [2].

Polynomial basis representation is widely accepted since the organizations which set the boundaries and limits for cryptography adopt polynomial basis as the choice. Various architectures have been used to realize polynomial basis multipliers like bit parallel, bit-serial, digit serial, semi-systolic, systolic, pipelined, and sequential. Although the bit-parallel multiplier consumes more area, it performs faster computation than other multipliers. Systolic multiplier architecture has a complete network of identical symmetrical elements and is preferred to perform fast computations [3]. These homogenous elements in systolic architecture are

* Corresponding author.
E-mail addresses: sakshidevi.phdece17@pec.edu.in (S. Devi), ritamahajan@pec.edu.in (R. Mahajan).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

known as cells or nodes. Each node performs highly complex operations.

Encryption schemes like AES, RSA, NTRU, Torus-based cryptography depend on finite field arithmetic High speed encryption is a must to ensure efficient data security. To accomplish fast computations, a bit parallel systolic polynomial basis multiplier has been proposed in this paper. The use of cut-set retiming in the multiplier design can significantly improve the critical path delay of the bit parallel systolic multipliers. The proposed multiplier is efficient in terms of area and time complexity. It has got the best area-delay tradeoff and therefore, it can be extended to work for other specific polynomials like trinomials, pentanomials, etc.

The main contributions of the paper are as follows:

a) A modular multiplication algorithm based on the conventional modular multiplication and the modified interleaved modular multiplication algorithm is formulated to find the product: P*Q mod T.
b) A Signal Flow Graph (SFG) has been represented which makes use of various types of cells to find the bitwise product of two elements P and Q.
c) Cut-set retiming has been performed on the SFG to obtain the low critical path delay based systolic multiplier and a bit-parallel multiplier architecture has been derived from it which has improved area and time complexity.

The rest of the paper is organized as follows: Section 2 describes the literature of bit parallel polynomial basis systolic multipliers. Section 3 addresses the fundamentals of the theoretical bit parallel systolic polynomial multiplier. The algorithm for a bit-parallel systolic multiplier of the polynomial basis is proposed in section 4 along with the description of multiplier architecture. The results and discussions of the proposed multiplier have been described under Complexity and Comparison in section 5. The FPGA implementation results of the proposed multiplier are compared with the earlier best-reported bit parallel polynomial basis systolic multipliers for m = 8 and m = 163 in sub-section 5.1. The conclusion is stated in section 6.

2. Related work

For modular multiplication in finite field arithmetic, a variety of multipliers have been suggested. Mastrovito multiplier performs parallel multiplication [5]. Multipliers based on the first LSB and MSB algorithms are used to perform the multiplication starting from the operand's least significant and most significant bit respectively [6,7]. A digit serial multiplier with multiplier A(x), multiplicand B (x) and the irreducible polynomial P(x) is partitioned in digits and iteratively processed to do modular multiplication [8]. Several serial-in serial-out and parallel-in parallel-out multipliers have been proposed [9–11]. Software implementation of the Montgomery multiplier is given in [12]. An area and time-efficient Montgomery algorithm is given in [13] which is time-independent. Mastrovito multiplier-like matrix–vector product techniques have high space complexity [14]. Montgomery multiplier can be used to multiply shifted polynomial basis numbers [15].

To decrease the multipliers' space complexity, numerous techniques are used like linear feedback shift register-based polynomial basis multiplier [16], multiplexer-based modular multiplier [17], etc. Using traditional multiplication and a folded method, modular multiplication can be rendered time-independent [18]. All these multipliers are designed to work only with fixed GF(2^m) fields. To solve this problem, versatile architectures that support arbitrary GF(2^m) fields are given in [19]. Irreducible pentanomial based bit parallel polynomial basis multiplier has less space and time complexity [20]. A bit-parallel and a bit-serial systolic array multiplier based on LSB first algorithms have been presented in [21]. More optimized architecture for Montgomery multiplication with reduced area and space complexity over [13] is proposed in [22]. The dual basis digit serial multiplication can be decomposed into sub-multiplication units to incorporate the parallel processing which will further reduce the latency of the multiplier [23]. The hardware accelerator can be made with high performance which can perform large-scale matrix multiplication [24]. Modified interleaved modular multiplication is employed to reduce the space and time complexity of all irreducible polynomials [25].

Karatsuba algorithm and Toeplitz matrix–vector can be employed to propose modular multipliers over irreducible trinomials [26,27]. The area-delay product can be improved in a Karatsuba multiplier by employing a reordered normal basis [28]. The authors in [29] have achieved the efficient area and low latency systolic multiplier. However, it is designed specifically for trinomials and gives good performance for even 'm' over GF(2^m). The latency of the multiplier can be improved when implemented in digit serial form [30]. A low-cost n-term Karatsuba algorithm-based multiplier has been proposed in [31]. The reliability of the bit-parallel multipliers can be improved by the parity-based concurrent error detection technique [32]. The linear feedback shift register can be used to reduce the area complexity in bit-serial polynomial basis multipliers [33]. A zero-suppressed binary decision diagram [34] improves the speed of the GF multipliers. The synthesis tool for FPGA implementation of the multiplier can be provided with freedom for optimization to achieve low delay [35].

A bit parallel systolic polynomial basis multiplier with reduced critical path delay is proposed in this paper. So, the total delay of the multiplier gets reduced which increases the speed of operation of the multiplier and therefore, the best tradeoff between area and delay is obtained by the proposed multiplier according to the author's knowledge. The results have been verified for m = 8 and 163.

3. Theory of the problem

A set of elements is called a finite field if it forms the additive group, multiplicative group and also holds the distributive law. The number of items in the set is called field order. Finite fields are divided into two types: Prime fields (GF(a), a is a prime number) and Extension fields (GF(2^m)). In the prime field, the addition and multiplication are done modulo 'p' whereas, in the case of extension fields, addition modulo 2 is simply bitwise XOR, and multiplication is done by dividing the product of two polynomials by irreducible polynomial and considering only the remainder [4].

Let T(x) be an m degree irreducible polynomial defined in the finite field over GF(2^m) as follows:

$$T(x) = x^m + t_{m-1}x^{m-1} + \ldots\ldots t_2x^2 + t_1x^1 + t_0 \tag{1}$$

where $\{t_i \in \mathrm{GF}(2), \text{for } 0 \le i \le (m-1)\}$.

The set $\{1, \beta, \beta^2, \beta^3, \cdots\cdots\cdots\cdots \beta^{m-1}\}$, where β is a root of T(x), is known as the polynomial basis and is used to denote the elements of the field. Let P and Q be two field elements defined in GF (2^m). The representation of P and Q in polynomial form is done as follows:

$P = \sum_{i=0}^{m-1} p_i\beta^i$ and $Q = \sum_{i=0}^{m-1} q_i\beta^i$ where p_i and $q_i \in \{1,0\}$, for i = 0,1, 2..., $(m$-2), $(m$-1).

Using finite field arithmetic, the multiplication of P and Q over GF (2^m) is performed as follows:

$$R = P.QmodT(\beta) = \sum_{i=0}^{m-1} r_i\beta^i = (r_0 + r_1\beta + r_2\beta^2 + ... + r_{m-1}\beta^{m-1}) \Rightarrow R$$
$$= \left(\sum_{i=0}^{m-1} q_i\beta^i\right).PmodT(\beta) \Rightarrow R$$
$$= (q_0 + q_1\beta + q_2\beta^2 + ... + q_{m-1}\beta^{m-1})PmodT(\beta)$$

$$\Rightarrow R = q_0PmodT(\beta) + q_1\beta PmodT(\beta) + q_2\beta^2PmodT(\beta) + ... + q_{m-1}\beta^{m-1}PmodT(\beta)$$

$$\Rightarrow R = q_0P_0 + q_1P_1 + q_2P_2 + ... + q_{m-1}P_{m-1} \tag{2}$$

where $P_0 = P \bmod T(\beta)$, $P_1 = \beta P \bmod T(\beta)$, $P_2 = \beta^2 P \bmod T(\beta)$, $P_{m-1} = \beta^{m-1} P \bmod T(\beta)$

and in general,

$$P_i = \beta^i PmodT(\beta) \tag{3}$$

In equation (3), the P_i gives the modular reduction value for an ith iteration. This modular reduced value is multiplied with q_i for ith iteration with the help of AND operation. This way is followed to obtain the product for all the iterations recursively and finally, these product terms are added together to produce R. The values of P_i s' can be obtained as follows:

For i = 0,

$$P_0 = PmodT(\beta) = (p_0 + p_1\beta + p_2\beta^2 + ... + p_{m-1}\beta^{m-1}) \; mod(t_0 + t_1\beta + t_2\beta^2 + ... + t_{m-1}\beta^{m-1} + \beta^m) \tag{4}$$

In Eq. (4), the degree of P is less than T and therefore, no modular reduction is performed. Hence, $P_0 = P$

For $i = 1$,

$$P_1 = \beta PmodT(\beta) = \beta(p_0 + p_1\beta + p_2\beta^2 + ... + p_{m-1}\beta^{m-1}) \; mod(t_0 + t_1\beta + t_2\beta^2 + ... + t_{m-1}\beta^{m-1} + \beta^m)$$

$$\Rightarrow P_1 = (p_0\beta + p_1\beta^2 + p_2\beta^3 + ... + p_{m-1}\beta^m) \; mod(t_0 + t_1\beta + t_2\beta^2 + ... + t_{m-1}\beta^{m-1} + \beta^m)$$

$$\Rightarrow P_1 = p_{m-1}t_0 + (p_{m-1}t_1 + p_0)\beta + (p_{m-1}t_2 + p_1)\beta^2 + ... + (p_{m-1}t_{m-1} + p_{m-2})\beta^{m-1} \tag{5}$$

and so on.

All the multiplications like $p_{m-1}t_0$,$p_{m-1}t_1$, $p_{m-1}t_2$... and so on are performed using AND gate. Likewise, all the mod 2 additions such as $(p_{m-1}t_1+p_0)$, $(p_{m-1}t_2+p_1)$ .... and so on are performed using XOR operation in Eq. (5) since, in finite field arithmetic, the modulo 2 addition is simple the XOR operation. When comparing Eqs. (4) and (5), it is observed that P is left-shifted (p_{m-2}, p_{m-3} , ... p_2 , p_1 ,p_0) followed by the addition of this left-shifted P-value to the bit-wise product p_iT ($p_{m-1}t_{m-1}$, $p_{m-1}t_{m-2}$ ... p_{m-1} t_2,$p_{m-1}t_1$, $p_{m-1}t_0$) to produce the final result R. This concept forms the basis of finite field multiplication.

4. Bit parallel architecture of proposed polynomial basis systolic multiplier

According to the conventional modular multiplication process as described in section 3 and the modified interleaved modular multiplication algorithm mentioned in [25], the modular multiplication of two polynomials: P and Q, each with the degree (m-1) over GF(2^m) can be computed as shown in algorithm 1.T is the irreducible polynomial with degree 'm' and R_m is the final output. R_0 is initialized as $[000......00]_{m\ bit}$. For the 'm' bit multiplier, the product q_jP_j is calculated 'm' times.

The recursive computations of q_jP_j from Eq. (2) are performed by steps 2–6 of algorithm 1. As observed from equation (3), the value of P_j in every jth iteration is calculated by multiplying it with β^j followed by its modular reduction using T(β), which is expressed in steps 4–5. We do not require P_m, therefore, the loop is iterated 'm-1'times and final output R_m is calculated outside the loop to save the resources.

```
Algorithm 1: Modular multiplicatin
Input: $P$,$Q$,$T$ (irreducible polynomial)
Output: $R_m$ = ($P \times Q$) and $T$
1:Intialize:$R_0$ = $[0000000]_m$
2:for $j$ = 0 to $m$-2 do
3:   $R_{j+1}$=$R_j \oplus (P_j \cdot q_j)$          //If $q_j$ = 1, then $R_{j+1}$=$R_j \oplus P_j$ else
     $R_{j+1}$=$R_j$
4:   $\hat{P}_j$=$P_j$ < <1          //$P_j$ is left-shifted by 1 bit to obtain $\hat{P}_j$
5:   $P_{j+1}$=$\hat{P}_j \oplus (T \cdot pj[m-1])$          //If the msb of $P_j$ is '1', then
     $P_{j+1}$=$\hat{P}_j \oplus T$, else $P_{j+1}$=$\hat{P}_j$
6:end for
7:$R_m$=$R_{m-1} \oplus (P_{m-1}$,$q_{m-1})$
```

In algorithm 1, R_{j+1} stores the intermediate partial products of $P_j.q_j$, where P_j's are obtained using equation (3) and then, P_j's are multiplied to q_j according to Eq. (2). Step 3 of algorithm 1 performs the bit multiplication and field addition step. The left-shifted P values are added to the product p_iT as described in Eq. (5) for j = 1, which are calculated in steps 4–5 of algorithm 1. In other words, steps 4–5 of algorithm 1 are responsible for the modular reduction and perform the operation P_{j+1} = $P_j.\beta$ mod T. Algorithm 1 makes use of two logic gates: XOR gate and AND gate to perform the logic operations in the multiplier.

The recursive implementation of bit parallel systolic multiplier architecture is performed in three steps: modular reduction, bit multiplication and field addition. The signal flow graph (SFG) for this recursive multiplication is depicted in Fig. 1. SFG includes 'm' modular reduction cells, 'm' bit multiplication cells and 'm' field addition cells. The functions of the bit multiplication cell (W cell), field addition cell (Y cell) and modular reduction cell (Z cell) are shown in Fig. 2. Each W cell performs the multiplication of a single bit of Q, starting from the LSB, with the reduced form of P as depicted in Fig. 2(a). Y cell, shown in Fig. 2(b), is used to perform addition according to step 3 of the multiplication algorithm. Z cell is used to reduce the degree of P modularly by one and performing step 4-5 of algorithm 1 as shown in Fig. 2(c). For example, W(0) cell gives the product (P_0 . q_0), Y(0) cell gives field addition output R_1 from inputs R_0 and $P_0.q_0$ and Z(0) cell gives modular reduced output P_1, in which firstly P_0 is left-shifted by 1 bit($\widehat{P}_0$) and then, this left shifted value $\widehat{P}_0$ is XORed with the product (T. p_0[m-1]). Similarly, R_{j+1} and P_{j+1} are calculated for m = 0 to m-2.

In other words, W(j) cell gives a product ($P_j.q_j$). Y(j) gives field addition output R_{j+1} from inputs R_j and $P_j.q_j$. Z(j) gives reduction output P_{j+1}. After the last iteration j = m-1, P_m is not calculated since it is not required further and only R_m is produced from step 7. In this way, all the iterations j = 0,1,2,3... (m-1) can be performed and the final output R_m can be obtained.

To decrease the critical path delay, cut-set retiming is performed on SFG as depicted in Fig. 3(a). The critical path delay of the proposed systolic multiplier is max{T_W + T_Y, T_Z) where T_W, T_Y, and T_Z are the critical path delays of W, Y and Z cells respectively. Therefore, T_A + T_X is the critical path delay(obtained from T_Z) of the proposed multiplier, where T_A and T_X are the propagation delays of AND and XOR gate respectively.

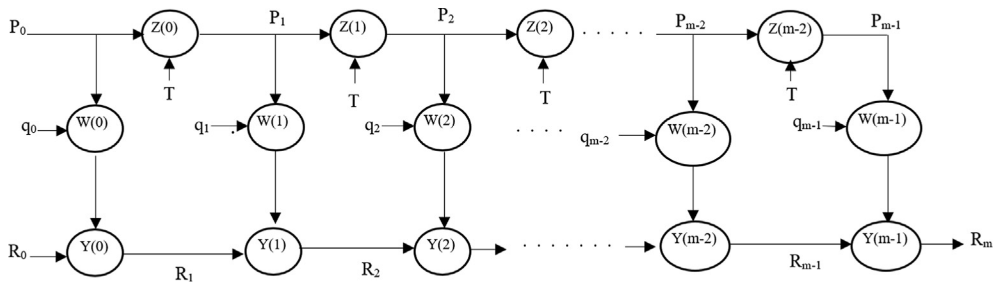
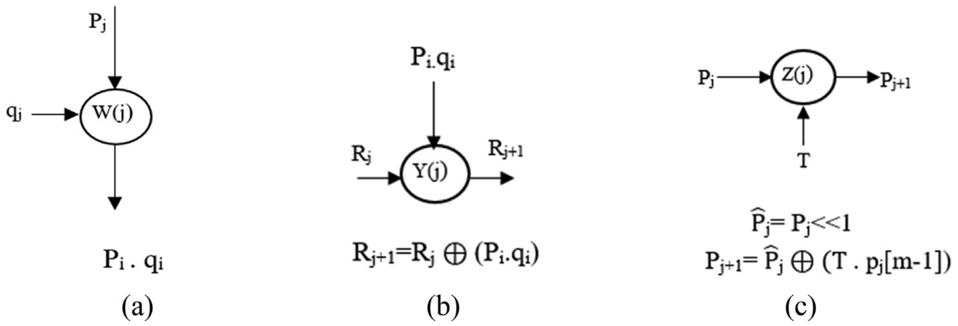


Fig. 1. SFG for the proposed multiplier.



$P_i \cdot q_i$
$R_{j+1}=R_j \oplus (P_i.q_i)$
$\widehat{P}_j = P_j << 1$
$P_{j+1} = \widehat{P}_j \oplus (T \cdot p_j[m-1])$
(a)
(b)
(c)

Fig. 2. (a) Logic function of W(j) block, (b) Logic function of Y(j), (c) Logic function of Z(j) block.

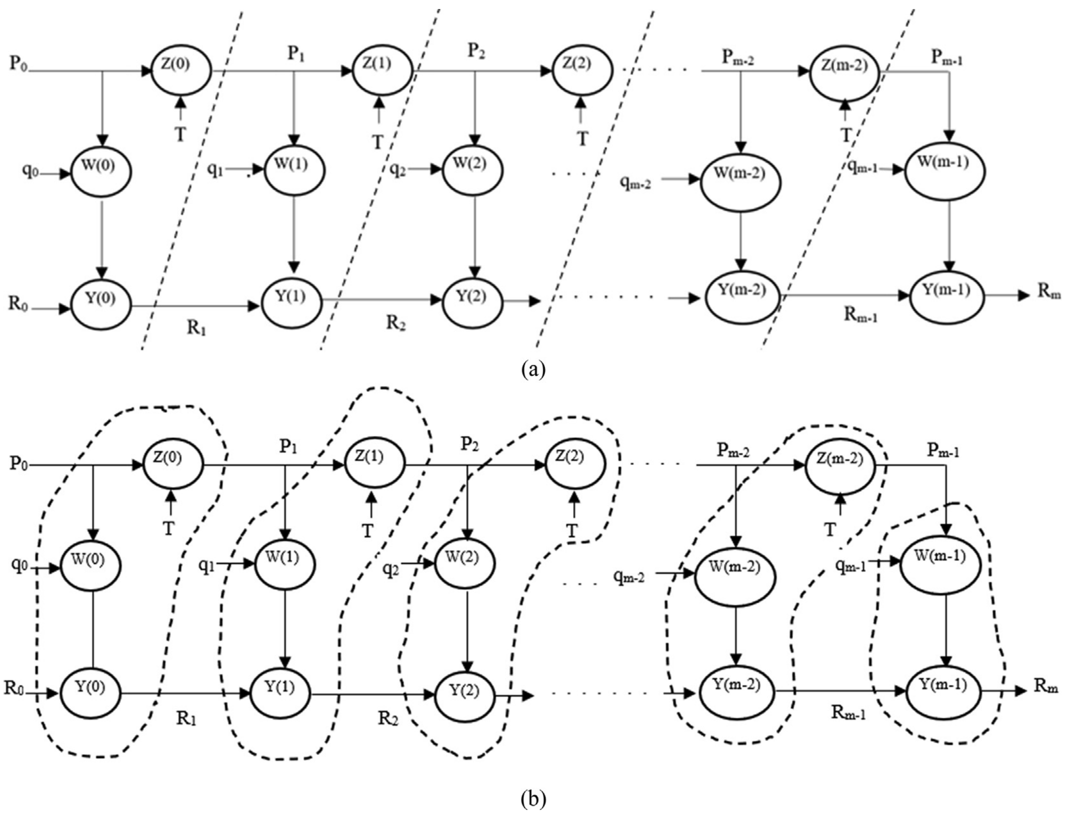


Fig. 3. Cut-set retiming of the SFG (a) Cut-set formation, (b) Processing elements construction from SFG blocks.

It is clear that computation of R takes place ‘m’ times and hence, the complete multiplication process is divided into ‘m’ processing elements or nodes after performing cut-set retiming on SFG as shown in Fig. 4(a). Each processing node performs the bit multiplication, addition and modular reduction operations except the last node, which performs only the multiplication and addition as shown in Fig. 4(b) and (c) respectively.

To make the multiplier more scalable and regular, these processing nodes are further divided into K cells to perform bit-wise AND (.) and XOR ($\oplus$) operations. The processing elements from 0

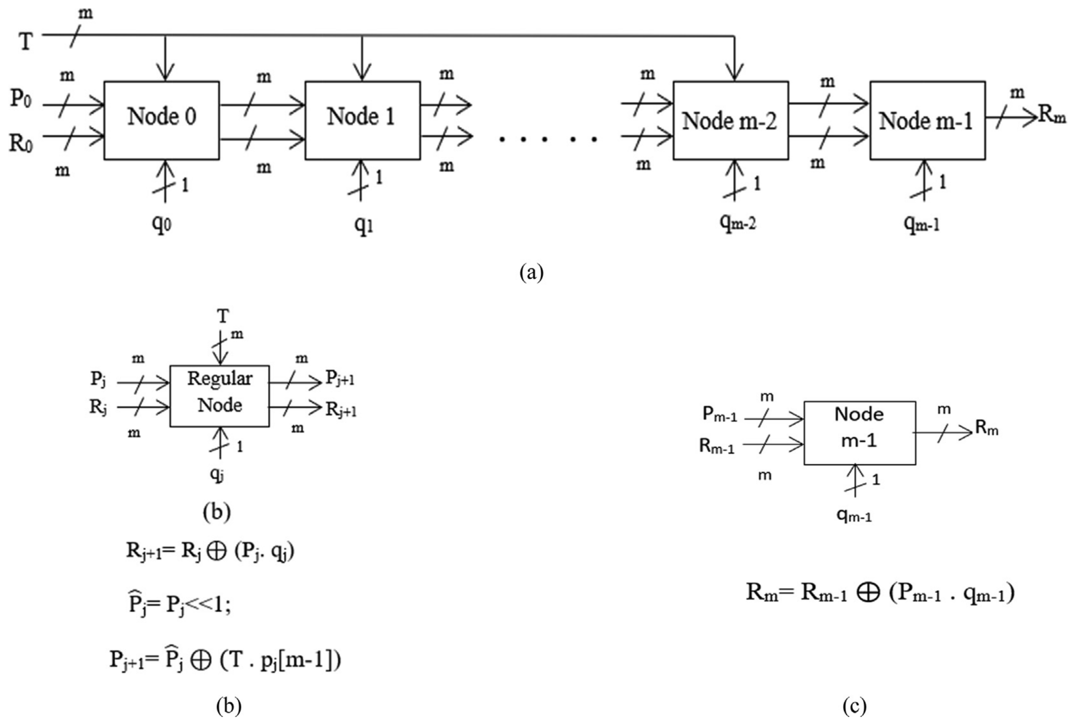


Fig. 4. Systolic multiplier using Nodes (a) systolic multiplier, (b) Regular nodes from 0 to [m-2], (c) Node (m-1).

to [m-2] are divided into ‘$2m^2$-2m’ K cells which include (m^2-m) cells for multiplication plus field addition and (m^2-m) cells for modular reduction, whereas the last processing element [m-1] is divided into ‘m’ K cells which include multiplication only as shown in Fig. 5. The first group of ‘m^2’ K cells from the upper block represents the multiplication and addition operations and calculates R_{j+1} of step 3 (or R_m of step 7) in algorithm 1. The group of ‘m^2-m’ K cells in the lower block represents the modular reduction operation and finds P_{j+1} according to steps 4–5 of algorithm 1. Each K cell includes one AND gate and one XOR gate as represented in Fig. 6.

Due to the pipelining of registers in Fig. 5, the critical path delay is $T_A + T_X$, where T_A and T_X are the delays of an AND gate and an XOR gate. From Fig. 5, the utilization of the number of AND gates, XOR gates, and latches are ($2m^2$-m), ($2m^2$-m) and m^2 respectively. It may be noted that 1-bit latches are required to store the intermediate R_j values. The advantage of the proposed multiplier is that it has low critical path delay and therefore, it is faster in operation. The final output is produced in each clock cycle taking m clock cycles initially. Being regular and modular, it can be scaled to high m valued multipliers. The proposed bit parallel multiplier architecture is important because of its systolic structure which makes it suitable for VLSI implementations. However, due to its high space complexity, it does not fit into low-area applications.

5. Complexity and Comparison

5.1. Results

The proposed K cell-based polynomial basis systolic multiplier is compared, in terms of the area and time complexity with the available systolic multipliers as shown in Table 1. NIST has recommended five irreducible polynomials that are m = 163,233,283,409 and 571. F(x) = $x^8 + x^4 + x^3 + x + 1$ (for m = 8) and F(x) =$x^{163} + x^7 + x^6 + x^3 + 1$ (for m = 163) are used in Table 2 to compare the area and time complexity of systolic multipliers. The proposed multiplier occupies ‘$2m^2$-m’ 2-input AND gates, ‘$2m^2$-m’ 2-input XOR gates and m^2 1-bit latches.

To evaluate the space complexity as shown in Table 1, traditional CMOS logic is employed which includes 4 transistors to make a 2-input XOR gate, 6 transistors each for 2-input AND gate and a 1-bit 2:1 multiplexer,8 transistors for a 1-bit latch, 8 transistors for 3-input XOR gate. ST microelectronics' real-time existing circuits are used to find the delay in case of the proposed multiplier, where propagation delay of 2-input XOR gate (M74HC86) is 12 ns, 2-input AND gate(M74HC08) is 7 ns, 2:1 mux (M74HC257) is 11 ns, 4:1 MUX (M74HC153) is 16 ns and 3-input XOR gate is 24 ns since it is realized using two 2-input XOR gates (8 transistors). Table 2(for m = 8,163) depicts the area complexity of polynomial basis systolic multipliers as a function of the number of transistors and time complexity of polynomial basis systolic multipliers as a function of total delay which is equal to the product of latency and critical path delay. The 2:1 multiplexer in [25] is replaced by AND gate in the proposed multiplier to reduce the critical path delay which is the key factor for the reduction in the time complexity of the multiplier.

The proposed systolic bit parallel polynomial basis multiplier achieves the least time complexity as compared to the best reported systolic multipliers, for the same area complexity as observed from Table 2. To the best of our knowledge, our proposed multiplier has attained the least area-delay product due to a considerable decrease in the critical path delay, when compared with another bit parallel polynomial basis multipliers as presented in Fig. 7.

The proposed systolic multiplier and the multipliers available in [18,19] and [25] are modeled using HDL and implemented on the FPGA device Xilinx Virtex-7(XCV2000TFLG1925-2) using Xilinx Vivado 2019.1 verification tool. Table 3 lists the area occupied on the FPGA device in terms of the number of Logic cells, delay in ns, Power in W for m = 8,163. The area delay product is also compared for m = 8,163. It is evident from Table 3 that the multiplier shown in Fig. 5 takes less time to perform the computations as compared to the other multipliers with a slight decrease in the area while doing FPGA implementation.

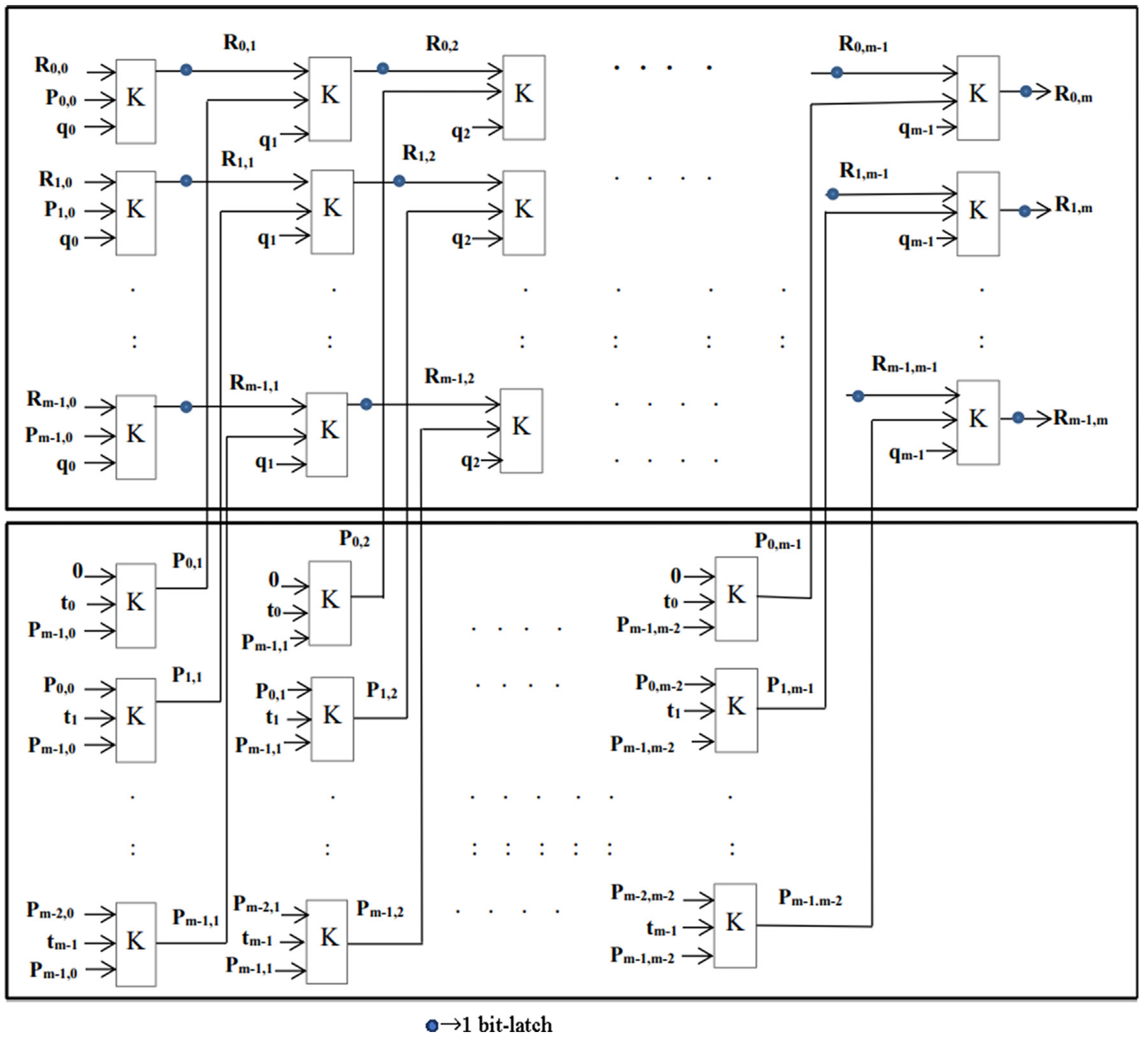


Fig. 5. Systolic multiplier using K cells for irreducible polynomials over GF(2 ^{m}).

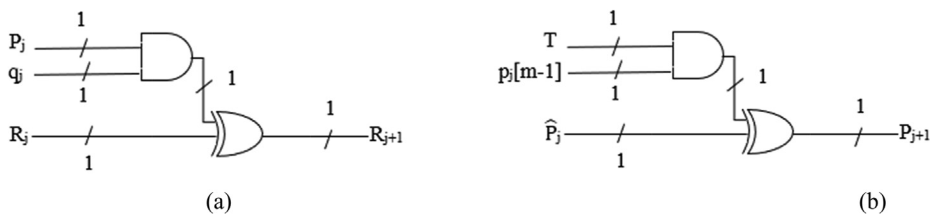


Fig. 6. Logic Diagram of K cell (a) For upper block, (b) For lower block.

Table 1
Area utilization and delay of polynomial basis systolic multipliers for irreducible polynomials over GF(2 ^{m}).

Design	No. of AND gates	No. of XOR gates	No. of multiplexers (2:1 Mux)	No. of latches	No. of clock cycles	Critical Path Delay
[16]	$2m^2$	$2m^2$	0	$3m^2$	m	$T_A + T_X$
[17]	m	$2m+(m^2/2)_t$	$(m^2 + m/2)_r$	$7m^2$	3m/2	$T_{4M} + T_{3X}$
[18]a	m^2	$m^2 + 2m$	0	$4m^2 + 3m$	3m	$T_A + T_X$
[18]b	m^2	m^2	0	$5m^2$	4m	$T_A + T_X$
[19]	m^2-m + 1	m^2-1	$2m^2$ + m-3	$2m^2$-m	2m	$T_A + T_x$
[21]	$2m^2$	$2m^2$	$2m^2$ + m-3	$7m^2$	3m	$T_A + T_x$
[25]	0	$2m^2$-m	$2m^2$-m	m^2	m	$T_M + T_x$
Fig. 5	$2m^2$-m	$2m^2$-m	0	m^2	m	$T_A + T_x$

t: 3input XOR gate, r: 4:1 Mux.

Table 2
Comparison of area and time complexity of the proposed systolic multiplier with the available systolic multipliers for m = 8 and m = 163.

Design	#Transistors		#Clock cycles (Latency)		Critical Path Delay (ns)	Total Delay(ns)		Area Delay Product	
	m = 8	m = 163	m = 8	m = 163		m = 8	m = 163	m = 8($\times 10^5$)	m = 163(($\times 10^{10}$)
[16]	2816	11,69,036	8	163	19	152	3097	4.28032	0.36205
[17]	5104	20,24,134	12	245	40	480	9800	24.4992	1.9836513
[18]a	2944	11,21,114	24	489	19	456	9291	13.42464	1.041627
[18]b	3200	13,28,450	32	652	19	608	12,388	19.456	1.645684
[19]	2352	10,08,302	16	245	19	304	4655	7.15008	0.4693664
[21]	5662	23,39,032	24	489	19	456	9291	25.81872	2.173195
[25]	1712	7,42,302	8	163	23	184	3749	3.15008	0.278289
Fig. 5	1712	7,42,302	8	163	19	152	3097	2.60224	0.229891

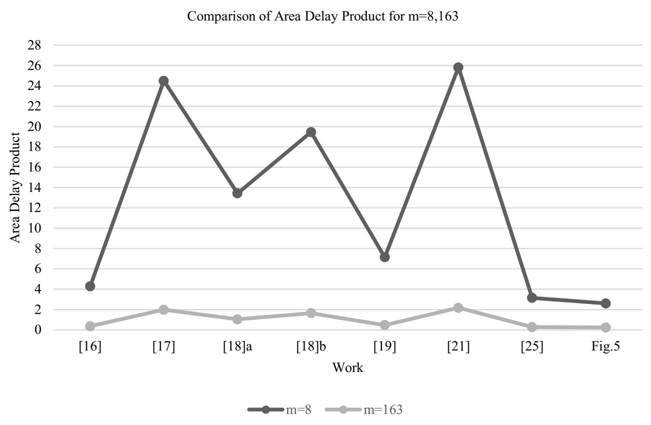


Fig. 7. Comparison of Area Delay Product of proposed multiplier with the multipliers available for m = 8163.

5.2. Discussions

The efficiency of the proposed multiplier is supported by the number of reductions achieved in the area and time complexities as shown in Table 2 for m = 8,163. The proposed multiplier achieves 27% less area than [19] and 69% less area than [21] for m = 8. The area complexity of our multiplier matches with the [25]. The critical path delay has been improved to the sum of one 2-input AND and 2-input XOR gate delay only due to pipelining of K cell registers. The time complexity of the proposed multiplier is 50% less than [19], 66% less than [21] and 17% less than [25] for m = 8. The area-delay product obtained by the proposed multiplier is 63% less than [19], 89% less than [21] and 17% less than [25]. The reduction in area and time complexity achieved by the proposed multiplier for m = 163 is approximately equal to the reductions achieved in the case of m = 8. Therefore, our multiplier has achieved the lowest time complexity and area-delay product. FPGA implementations of the proposed multiplier also show that the proposed multiplier is 90% delay efficient and 18% area efficient than [25] with a slight increase in the power consumption. Consequently, the area-delay product is the lowest as compared to the other multipliers.

The proposed bit parallel polynomial basis systolic multiplier has achieved the best area-delay tradeoff and therefore, it can be extended to other types of polynomial based multipliers like trinomials, pentanomials, equally spaced polynomials, etc. It performs parallel computations and is faster in operation as compared to its bit-serial counterparts. However, the number of resources(area complexity) utilized in the bit parallel multipliers is very high. It is limited to applications that give supreme priority to the speed and applications where the area utilization is not a major concern.

6. Conclusion

In this research work, the best tradeoff for the area and time complexity is obtained with the help of modification of the logic components of the processing elements of the polynomial basis systolic multiplier. It is observed that the proposed systolic polynomial basis multiplier over irreducible polynomials achieves 66% and 17% less time complexity due to low critical path delay for m = 8 and m = 163 when comparison analysis is made with the earlier best-reported multipliers available in the literature. The area complexity obtained by the proposed multiplier matches with the best results available. The area-delay product of the proposed multiplier is 89% and 17% less as compared to the existing multipliers. It is concurrent and therefore, it performs computations faster. Being regular, modular and scalar, it is better suited for VLSI implementations. The proposed low critical path delay multiplier can be optimized further to work for trinomials and pentanomials as well.

Table 3
Comparison of FPGA Implementation Results for m = 8 and m = 163.

Design	Total Delay		Area(#LCs)		Power(W)		ADP	
	m = 8 (ns)	m = 163 (μs)	m = 8	m = 163	m = 8	m = 163	m = 8(#LCs*ns*10^3)	m = 163(#LCs*μs*10^3)
[18]	363.36	7.403	190	1,54,635	0.771	3.6	69.038	1144.763
[19]	96.672	1.970	139	1,05,787	0.692	6.187	13.437	208.400
[25]	10.552	0.215	98	66,434	0.517	2.848	1.034	14.283
Fig. 5	1.004	0.0024	80	32,685	0.648	5.277	0.080	0.0784

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C. Paar and J. Pelzl, Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media, Nov 2009.
[2] Hsu IS, Truong TK, Deutsch LJ, Reed IS. A comparison of VLSI architecture of finite field multipliers using dual, normal, or standard bases. IEEE Trans Comput 1988;37(6):735–9.
[3] Kung HT. Why systolic architectures? IEEE Comput 1982;15(1):37–46.
[4] W. Stallings, Cryptography and network security: principles and practice, Upper Saddle River: Pearson, pp. 92-95,2017.
[5] Li Y, Ma X, Zhang Y, Qi C. Mastrovito form of non-recursive Karatsuba multiplier for all trinomials. IEEE Trans Comput 2017;66(9):1573–84.
[6] Song L, Parhi KK. Low-energy digit-serial/parallel finite field multipliers. J VLSI Sig Proc 1998;19(2):149–66.
[7] Jain SK, Song L, Parhi KK. Efficient semisystolic architectures for finite field arithmetic. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems 1998;6(1):101–13.
[8] Morales-Sandoval M, Díaz-Pérez A. Compact FPGA-Based Hardware Architectures for GF (2^ m) Multipliers. In: In 2013 Euromicro Conference on Digital System Design. p. 649–52.
[9] Yeh CS, Reed IS, Truong TK. Systolic multipliers for finite fields GF (2^m). IEEE Trans Comput 1984;33(4):357–60.
[10] Wang CL, Lin JL. Systolic array implementation of multipliers for finite fields GF (2/sup m/). IEEE Trans Circ Syst 1991;38(7):796–800.
[11] S.K. Jain and K.K. Parhi, "Low latency standard basis GF (2/sup m/) multiplier and squarer architectures," International Conference on Acoustics, Speech, and Signal Processing, Detroit, MI, USA, pp. 2747-2750, May 1995.
[12] Koc CK, Acar T. Montgomery multiplication in GF (2k), designs, codes and cryptography. Springer 1998;14(1):57–69.
[13] Chiou CW, Lee CY, Deng AW, Lin JM. Efficient VLSI implementation for Montgomery multiplication in GF (2m). Tamkang J Sci Eng 2006;9(4):365–72.
[14] Erdem SS, Yanık T, Koç ÇK. Polynomial basis multiplication over GF (2m). Acta Appl Mathemat Springer 2006;93(1-3):33–55.
[15] Fan H, Hasan MA. Relationship between GF (2m) Montgomery and shifted polynomial basis multiplication algorithms. IEEE Trans Comput 2006;55 (9):1202–6.
[16] Chiou CW, Lee CY, Lin JM. Finite field polynomial multiplier with a linear feedback shift register. Tamkang J Sci Eng 2007;10(3):253–64.
[17] Lee C-Y. Multiplexer-based bit-parallel systolic multipliers over GF (2m). Comput Electr Eng Elsevier 2008;34(5):392–405.
[18] Lee CY. Low complexity bit-parallel systolic multiplier over GF (2m) using irreducible trinomials. Integration VLSI J Elsevier 2008;41(1):106–12.
[19] Fournaris AP, Koufopavlou O. Versatile multiplier architectures in GF (2k) fields using the Montgomery multiplication algorithm. Integration VLSI J Elsevier 2008;41(3):371–84.
[20] Wu H. Bit-parallel polynomial basis multiplier for new classes of finite fields. IEEE Trans Comput 2008;57(8):1023–31.
[21] Kwon S, Kim CH, Hong CP. More efficient systolic arrays for multiplication in GF (2m) using LSB first algorithm with irreducible polynomials and trinomials. Comput Electr Eng Elsevier 2009;35(1):159–67.
[22] Kim K-W, Jeon J-C. Polynomial basis multiplier using cellular systolic architecture. IETE J Res 2014;60(2):194–9.
[23] Chiou CW, Lee C-Y, Lin J-M, Yeh Y-C, Pan J-S. Low-latency digit-serial dual basis multiplier for lightweight cryptosystems. IET Inf Secur 2017;11(6):301–11.
[24] Wei C, Song Y, Zhang D. A chain-multiplier for large scale matrix multiplication. In: In 2017 IEEE 12th International Conference on ASIC (ASICON). p. 792–5.
[25] Mathe SE, Boppana L. Low-power and low-hardware bit-parallel polynomial basis systolic multiplier over gf (2m) for irreducible polynomials. ETRI J 2017;39(4):570–81.
[26] Park SM, Chang KY, Hong D, Seo C. Low space complexity GF (2^ m) multiplier for trinomials using n-term karatsuba algorithm. IEEE Access 2019;7:27047–64.
[27] Pan JS, Lee CY, Sghaier A, Zeghid M, Xie J. Novel systolization of subquadratic space complexity multipliers based on Toeplitz matrix-vector product approach. IEEE Trans Very Large Scale Integr VLSI Syst 2019;27(7):1614–22.
[28] Xie J, Lee CY, Meher PK, Mao ZH. Novel Bit-parallel and digit-serial systolic finite field multipliers over $ GF (2^ m) $ based on reordered normal basis. IEEE Trans Very Large-Scale Integr (VLSI) Syst 2019;27(9):2119–30.
[29] Pillutla SR, Bopanna L. Area-efficient low latency polynomial basis finite field GF(2^m) systolic multiplier for a class of trinomials. Microelectron J 2020;97:1–8.
[30] Ibrahim A. Unified and scalable digit-serial systolic array for multiplication and division over GF (2m). IEEE Trans Comput Aided Des Integr Circuits Syst 2020;39(7):1546–9.
[31] Park S-M, Chang K-Y, Hong D, Seo C. Efficient bit-parallel multiplier for all trinomials based on n-term karatsuba algorithm. iEEE Access 2020;8:173491–507.
[32] Fei C, Fang Z, Ning W, Fen G, Jin W, Peiyao Q. A scalable bit-parallel word-serial multiplier with fault detection on GF (2^ m). In: In 2020 IEEE 20th International Conference on Communication Technology (ICCT). p. 1660–4.
[33] Imana JL. LFSR-based bit-serial $ GF (2^m) $ GF (2^m) multipliers using irreducible trinomials. IEEE Trans Comput 2020;70(1):156–62.
[34] Ito A, Ueno R, Homma N. Efficient formal verification of galois-field arithmetic circuits using ZDD representation of boolean polynomials. IEEE Trans Comput Aided Des Integr Circuits Syst 2021.
[35] J.L. Imana, "Low-Delay FPGA-Based Implementation of Finite Field Multipliers," in IEEE Transactions on Circuits and Systems II: Express Briefs, 2021.