Middle-East Journal of Scientific Research 21 (11): 2064-2071, 2014 ISSN 1990-9233 © IDOSI Publications, 2014 DOI: 10.5829/idosi.mejsr.2014.21.11.21795

# Low Complexity Multiplier for GF (2<sup>m</sup>) Based All One Polynomial

M. Anto Bennet, M. Manimaraboopathy, P. Maragathavalli and T.R. Dinesh Kumar

Department of ECE, VELTECH, Chennai-600062, India

**Abstract:** The area-time-efficient systolic structure for multiplication over GF (2m) based on irreducible all-one polynomial (AOP) and used a novel cut-set retiming to reduce the duration of the critical-path to one XOR gate delay. Basically, this paper is depends on digital electronics(ie.,logic gates)how to reduce the gate count.Finally it is used for what are techniques available in electronics(VLSI advanced technology).Here going to do is to reduce the power consumption,reduce the gate count,and to reduce the critical path in XOR gate in real time application.In input using the technique called register sharing an cut set retiming in that how to reduce the components and to get in area time efficient of systolic structure. The result obtained is in real time application of security purposes for example ATM,etc., to get the area time efficient systolic structure and security purposes in advanced VLSI technology. The application of the paper is mainly for security purposes and for irreducible polynomial of efficient implementation.

Key words: All-One Polynomial • Elliptic Curve Cryptography • System on Chip

### INTRODUCTION

Finite Field Multipliers over GF (2<sup>m</sup>) have wide applications in Elliptic Curve Cryptography (ECC) and Error Control Coding systems. Polynomial basis multipliers are popularly used because they are relatively simple to design and offer scalability for the fields of higher orders. Efficient hardware design for polynomialbased multiplication is therefore important for real-time applications. All-One Polynomial (AOP) is consists of binary number 0's and 1's considered suitable for systolic multipliers. The circuit complexitycan be reduced by using irreducible algorithm in each stage the number of gates are used that can be reduced at last obtained an single gate. Thereby, To achieve high throughput based on application. Thus the overall circuit complexity is reduced. All-one polynomial (AOP) is one of the classes ofpolynomials considered suitable to be used as irreducible polynomial for efficient implementation of finite field multiplication. Finite Field Multipliers over GF (2<sup>m</sup>) have wide applications in Elliptic Curve Cryptography (ECC) and Error Control Coding systems. Polynomial basis multipliers are popularly used because they are relatively simple to design and offer scalability for the fields of higher orders. Efficient hardware design for polynomial-based multiplication is therefore important for real-time applications. The design of systolic arrays is the

mapping of the algorithm to the processor array. However, not all algorithms can be systolized. Only highly regular algorithms with the structure of nested loops are suitable for systolic implementation. Systolic implementation of multiplication over  $GF(2^m)$  is usually very efficient in area-time complexity, but its latency is usually very large. Thus, two low latency systolic multipliers over  $GF(2^m)$ based on general irreducible polynomials and irreducible pentanomials. Systolic arrays have been designed for a wide variety of computationally intensive problems in signal processing. numerical problems, pattern recognition, database and dictionary machines, graph algorithms. Finite Field Multipliers have wide applications in Elliptic Curve Cryptography (ECC) and Error Control Coding systems. Polynomial basis multipliers are popularly used because they are relatively simple to design and offer scalability for the fields of higher orders. Efficient hardware design for polynomial-based multiplication is therefore important for real-time applications.

**Literature Survey:** BerkSunar (2004) had proposed a multiplier in convolution algorithm. This algorithm technique is to reduce the delay logarithmic in bit length. The advantages are reducing the complexities and area efficiency is high. The disadvantages is space complexity is high [1].Hanho Lee (2003) had proposed an architecture

called high speed Reed-Solomon(RS) decoder architecture using modified algorithm for fiber optic rates. This decoder implements 0.13m CMOS standard technology. The advantage is high speed data processing and detection and correction of errors. The disadvantage is critical path is high at a clock frequency [2].NeethuJohny and Binoy Joseph(2013) had proposed an finite field multipliers over GF (2<sup>m</sup>) used in technology like Elliptical Curve Cryptography(ECC) and Error Coding techniques. The advantage is to reduce the critical path in pipelining digital circuits and to reduce the time delay. The disadvantage is area efficient is little bit high [3]. ChiouYng Lee et al (2005) had proposed an Booth's algorithm using low complexity in dual basis multipliers. It saves about 9% space complexity. The advantage is parallel reduction of both space and time complexities. The disadvantage is multi bit processing is low in this process [4].Jean Claude et al (2010) had proposed an binary field multiplication representation in double polynomial system. It approach Fourier transform to perform reduction. The advantage is to avoid a multiplication required in Montgomery algorithm and in efficient method. The disadvantage is high complexity in this multiplier [5]. Henriquez (2003) had proposed an Galois field GF(2m) generated advantage in a space and time complexities. To reduce the multiplication by using irreducible polynomial in coding techniques. The advantage is to reduce the delay and complexities. The disadvantage is parallel multipliers in multilevel bit is less complexity [6]. YadollahEslami et al (2006) had proposed an Cryptography for secure purposes in electronic devices. It occupies small area; consume low power in this algorithm. The advantage is power consumption and less area delay. The disadvantage is multi bit of storage is less [7]. H.W. Leong et al (2002) had proposed a micro coded elliptic curve processor in FPGA technology. Using this technology to reduce the chip's i/o requirements. The advantage is control part of processor is micro coded in FPGA processor. The disadvantage is power consumption is high and cost is high [8].CancioMonterio et al (2013) had proposed a bit parallel multiplier over Galois field arithmetic algorithm in the circuit architecture. It implements the secure and low power dual logic circuit in bit parallel multiplier. The advantage is using CMOs technology to reduce the power consumption. The disadvantage is better security in high frequency rate [9]. Bimal Kumar Meher (2009) had proposed an finite field in efficient design of elliptic curve cryptography and error coding techniques for digital communication The advantage is security purposes and

cost effective is low. The disadvantage is less efficient during communication in elliptic curve cryptography [10]. Ashutosh Kumar Singh *et al* (2009) had proposed an error tolerant hardware efficient in very large scale integration architecture for bit parallel systolic multiplication. The advantage is operate in both dual base and polynomial base in efficient manner. The disadvantage is cost is high effective and space complexity is high [11]. Kazutoshi Wakabayashi *et al* (2000) had proposed an System on Chip (SOC) design method and flow from the view points in electronic application system. The advantage is chip is reduced in electronic devices and power consumption is also reduced. The disadvantage is physical design in Soc of electronic devices is little bit difficult [12].

#### MATERIALS AND METHODS

**Processing Elements (PE[0], PE[1], PE[m+1], Regular PE):** The structure of PE [0] is shown in Fig.1. It consists of an AND cell and a BSC. Each XOR cells and AND cells in the PE consists of (m+1) number of gates working in parallel. The regular PE, as shown in Fig.2, consists of three basic cells, e.g., the bit-shift cell (BSC), the AND cell and the XOR cell. The PE[m+1] of the systolic structure in Fig.5.3 consists of only an XOR cell, which performs bit-by-bit XOR operations of its pair of m-bit inputs

**Delay Unit:** In delay units are using D- Flip flop. The D flip-flop tracks the input, making transitions with match those of the input D. The D stands for "data"; this flip-flop stores the value that is on the data line. It can be thought of as a basic memory cell. A D flip-flop can be made from a set/reset flip-flop by tying the set to the reset through an inverter. The result may be clocked.

**AC Unit:** Besides, an Addition-Cell (AC) is required to perform the final addition of the outputs of the two systolic arrays, as shown in Fig 4. It performs the XOR operation.

**BSC (Bit Shift Cell):** The bit shifts are sometimes considered bitwise operations, because it operates only the binary representation of an integer instead of its numerical value; however, the bit shifts do not operate on pairs of corresponding bits and therefore cannot properly be called bit-wise. In these operations the digits are moved, or shifted, to the left or right. The BSC in the PE performs the bit-shift operation according to



Fig. 1: PE[0]



Fig. 2: Regular PE



Fig. .3: PE [m+1]

$$u \to v$$
 AC  $w w \leftarrow u + v$ 

Fig. 4: AC unit

$$A^{i+1=}a_0^{i+1} + a_0^{i+1} + a_1^{i+1} \cdot \alpha + \dots + a_m^{i+1} \cdot \alpha$$
 (1)

Systolic Structure: A systolic is said to be reversible if there is a one-to-one and onto mapping between the vectors of inputs and outputs; thus the vector of inputs can be always reconstructed from the vector of outputs. Thus, the number of outputs in a reversible gate or circuits has to be the same as the number of inputs. Output functions of binary reversible logic gates equal to 1 for exactly half their input assignments are called balanced. Logic design of reversible circuits is quite different from designing conventional irreversible logic circuits. In reversible circuits have to use at least one gate is used to duplicate a signal. Moreover, for realization of non balanced Boolean functions with a reversible circuit, it is necessary to add constant signals to input of circuits. A systolic array formed by interconnecting a set of identical data-processing cells in a uniform manner is a combination of an algorithm and a circuit that implements it and is closely related conceptually to arithmetic pipeline.

In a systolic array, data words flow from external memory in a rhythmic fashion, passing through many cells before the results emerge from the array's boundary cell and return to external memory. The external memory connected to the systolic array's boundary cell stores



Fig. 5: Systolic Multiplier



Fig. 6: Low Latency Systolic Multiplier

both input data and results. The underlying principle of systolic array is to achieve massive parallelism with a minimum communication overhead and generally speaking, a systolic array is easy to implement because of its regularity and easy to reconfigure because of its modularity. The classical logic synthesis methods can be used, but they generate too many number of gate output signals, making the circuit extremely complex. The basic design of systolic multiplier thus derived is shown inFig.5. It consists of (m+2) PEs and the functions of the PEs are shown in Fig 5. During each cycle period, the regular PE (from PE [2] to PE [m - 1]) not only performs the modular reduction operation. But also performs the bit-multiplication and bit-addition operations concurrently.

**Low Latency Systolic Architecture:** For irreducible AOP, m is an even number. Therefore, let l and P be two integers such that (m+1) = lP+r, where r is an integer in the range 0 = r = l. For example, if P=m/2, then l=2, r=1 can be rewritten as

$$C = \sum_{i=0}^{m/2} X_i + \sum_{i=\frac{m}{2}+1}^{m} X_i$$
(2)

One of the sum contains [(m/2) +1] partial products while the other has m/2 partial products. The systolic structure of Fig. 6 could be modified to a form shown in Fig. 6(a), which consists of two systolic branches. The upper branch consists of [(m/2) +2]PEs and the lower branch consists of (m/2+1) PEs and a delay cell. Besides, an Addition-Cell (AC) is required to perform the final addition of the outputs of the two systolic arrays, as shown in Fig.6 (b). It is observed that the two systolic



Fig. 7: Low-latency register-sharing systolic structure. (a) The systolic Structure. (b) Structure of PE [1]. (c) Structure of a regular PE (from PE [2] to PE [m/2-1]). (d) Structure of PE [m/2]. (e) Structure of PE [m/2+1].

branches in Fig. 5.6 share the same input operand and the PEs in both the branches perform the same operation except the last PE in each of the branches.

The proposed structure (Fig.7) requires [(m/2)+2] PEs and one AC. Each of the regular PEs consists of 2(m+1) XOR gates in a pair of XOR cells and 2(m+1) AND gates in a pair of AND cells. Besides, the AC requires (m+1) XOR gates.

These input undergone polynomial multiplication to do polynomial multiplication, the max no. of multipliers are used in order to reduce the no of multipliers by using irreducible algorithm. By this algorithm, No of inbuilt gates to frame an single multiplier have been reduced from 0 to m/2+1 level

#### RESULTS

The AND operation is used to perform multiplication of two or more inputs. If the inputs are 1,then it produce the result '1'.Otherwise it produce output '0'.is shown in Fig 8.

The Bit Shift Cell is used to shift the input to right by 1 bit position is shown in Fig 9.

The XOR operation is used to produce the result 1 if the inputs are different and produces result 0 if the inputs are same is shown in 0ig 10.

The processing element0 (PE0) produces the corresponding output to the given input and binary digit 0 or 1 by calling the function which is inbuilt in it to perform he specific operation is shown in Fig 11.

The processing element 1 (PE1) produces the corresponding output to the output produced by PE0 and binary digit 0 or 1 by calling the function which is inbuilt in it to perform he specific operation is shown in Fig 12.

The regular PE produces the output to the multiple inputs by comparing it with the binary digit 0 or 1 by calling the function which is inbuilt in it to perform he specific operation corresponding to the input given is shown in Fig 13.

The PE4 produces the corresponding output to the input and binary digit 0 or 1 by calling the function which is inbuilt in it to perform he specific operation and it is added to the additional cell to perform EX-OR operation is shown in Fig 14.

The M=6 systolic structures produces the output to the multiple inputs by comparing it with the binary digit 0 or 1 by calling the function which is inbuilt in it to perform he specific operation corresponding to the input given is shown in Fig 15.

The low latency systolic structure produces the corresponding output to the input and binary digit 0 or 1 by calling the processing element which consists of function which is to be performed is inbuilt in it to perform the specific operation is shown in Fig 16.

The Register Sharing multiplier systolic produces the output to the multiple inputs by comparing it with the binary digit 0 or 1 by calling the function which is inbuilt to perform the specific operation and it is added to the additional cell to perform EX-OR operation. The Register Sharing multiplier mainly used to minimize the register requirement is shown in Fig 17.

Middle-East J. Sci. Res., 21 (11): 2064-2071, 2014



	Messages							
<b>□-</b> ◆ <i>M_S</i> <b>□-</b> ◆ <i>M_S</i>	ystolic_Multipi	0000101 1000010	0000101					
- 18 A	Now	300 ns	111111	200 ns	 400 ns	600 ns	800 ns	100
£ 2 9	Cursor 1	0 ns	0 ns					

Fig. 9: Bit Shift Cell

Messages		
± ♦ /VAC/A	10011	10011
⊕ ♦ /VAC/B	01100	01100
	11111	
≗≣⊛ Now	200 ns	
🔓 🖉 🗧 Cursor 1	0 ns	Ons



Messages			
/M_Systolic_Multiplier/PE0/A /M_Systolic_Multiplier/PE0/P	1111110 St0	1110001	1111110
	0111111	1111000	0111111
M_Systolic_Multiplier/PE0/Y	0000000	1110001	0000000
M_Systolic_Multiplier/PE0/M_PE10/A	1111110	1110001	1111110
M_Systolic_Multiplier/PE0/M_PE10/B	0111111	1111000	0111111
/M_Systolic_Multiplier/PE0/M_PE20/SIn	St0		
M_Systolic_Multiplier/PE0/M_PE20/MIn	1111110	1110001	1111110
M_Systolic_Multiplier/PE0/M_PE20/MOut	0000000	1110001	0000000
Now Now	700 ns		
🔓 🌽 Cursor 1	0 ns	0 ns	

Fig. 11: Processing Element0 (PE0)

Messages	1 5			
/M_Systolic_Multiplier/PE1/A /M Systolic Multiplier/PE1/P	0111111 Sti	11111000	0111111	
/M_Systolic_Multiplier/PE1/8	0000000	1110001	20000000	
M_Systolic_Multiplier/PE1/X	1011111	0111100	1011111	
M_Systolic_Multiplier/PE1/Y	0111111	1111000	0111111	
M_Systolic_Multiplier/PE1/Z	0000000	1110001	0000000	
M_Systolic_Multiplier/PE1/M_PE10/A	0111111	1111000	0111111	
M_Systolic_Multiplier/PE1/M_PE10/B	1011111	0111100	1011111	
/M_Systolic_Multiplier/PE1/M_PE20/SIn	St1			
M_Systolic_Multiplier/PE1/M_PE20/MIn	0111111	1111000	0111111	
M_Systolic_Multiplier/PE1/M_PE20/	0111111	1111000	0111111	
Now Now	700 ns			
🚔 🖉 Cursor 1	0 ns	0 ns		

Fig. 12: The Processing Element 1 (PE1)

Messages				
M_Systolic_Multipli	1011111	0111100	1011111	
•	0111111	1111000	10111111	
M_Systolic_Multipli	0000000	1110001	20000000	
/M_Systolic_Multipli	St1			
/M_Systolic_Multipli	1101111	0011110	1101111	
/M_Systolic_Multipli	1011111	0111100	1011111	
H_ /M_Systolic_Multipli	0111111	0001001	0111111	
M_Systolic_Multipli	1011111	0111100	1011111	
H-4 /M_Systolic_Multipli	1101111	0011110	1101111	
/M_Systolic_Multipli	St1			
M_Systolic_Multipli	1011111	0111100	1011111	
M_Systolic_Multipli	1011111	0111100	1011111	
Now Now	700 ns			
Cursor 1	0 ns	0 os		

Fig. 13: Regular PE

Messages			
🕰 🤣 /M_Systolic_Multiplier/PE6/A	1111101	1100011	1111101
M_Systolic_Multiplier/PE6/8	0001111	0101011	0001111
M_Systolic_Multiplier/PE6/C	0000000	0000000	
/M_Systolic_Multiplier/PE6/x	St0		
M_Systolic_Multiplier/PE6/U	0000000	0000000	
M_Systolic_Multiplier/PE6/W	0001111	0101011	20001111
/M_Systolic_Multiplier/PE6/PE2/SIn	St0		
M_Systolic_Multiplier/PE6/PE2/MIn	1111101	1100011	1111101
M_Systolic_Multiplier/PE6/PE2/MOut	0000000	0000000	
M_Systolic_Multiplier/PE6/PE3/A	0001111	0101011	0001111
M_Systolic_Multiplier/PE6/PE3/8	0000000	0000000	
M_Systolic_Multiplier/PE6/PE3/Out	0001111	0101011	0001111
Now Now	700 ns	ns 200 ns	400 ns 600 ns
Gurster 1		0.00	

Fig. 14: Processing Element4 (PE4)

Messages				
M_Systolic_Multiplier/A	-No Data-	1110101	0101010	1110000
M_Systolic_Multiplier/B	-No Data-	1000010	10 10 10 1	1110010
M_Systolic_Multiplier/C	-No Data-	0 1000 10	1100110	0010111
M_Systolic_Multiplier/X1	-No Data-	11111010	0010101	0111000
M_Systolic_Multiplier/X2	-No Data-	0111101	100 10 10	0011100
M_Systolic_Multiplier/X3	-No Data-	1011110	0100101	0001110
M_Systolic_Multiplier/X4	-No Data-	0101111	10 100 10	0000111
M_Systolic_Multiplier/X5	-No Data-	1010111	0101001	1000011
M_Systolic_Multiplier/X6	-No Data-	1101011	10 10 100	1100001
M_Systolic_Multiplier/X7	-No Data-	0000000	1010100	0000000
M_Systolic_Multiplier/Y1	-No Data-	1110101	0101010	1110000
M_Systolic_Multiplier/Y2	-No Data-	0000000		0111000
Now Now	800 ns			
Cursor 1	1568 ns			

Fig. 15: M=6 systolic structures

In Table No 1,Here mentioned the logic utilization of systolic structure. Basically slices occupied in logic gate very less only because then only reached the area time efficient implementation.The input is number of bonded used Input Output Bank(IOB) for the ports.

In Table No 2, The device ulitization of systolic structure consists of related logic and unrelated logic. In this low latency of systolic array have used additional

JTAG gate count for IOBs for reducing the period and latency.In giving the input of look up table for reducing the purpose of slices in fnite field multiplication.

In Table No 3, contains the register sharing technique that is mainly used for reduce the register components in the logic gates. Here the number of slices used for utilization of related logic is 200 i.e., the purpose of reduce the components in the processor, power consumption and get the area time efficiency of finite field implementation.

Middle-East J. Sci. Res., 21 (11): 2064-2071, 2014

Mowlatercy_syst#A         110001         110101         100010         110000         011101           Mowlatercy_syst#A         011101         100010         1000010         110000         011101           Mowlatercy_syst#A         0001001         110001         011101         011101         0111010           Mowlatercy_syst#A         000000         111001         0110001         0110001         0011001         0111000           Mowlatercy_syst#A         0000000         111001         0101010         1110000         0000000         111000           Mowlatercy_syst#A         0000000         111010         0101010         0101000         0000000           Mowlatercy_syst#A         0000000         111010         0101010         0101010         010000         0000000           Mowlatercy_syst#A         0011100         0000000         1110010         010000         010000         0000010           Mowlatercy_syst#A         001110         0000000         1100000         1100000         1100000         1100000           Mowlatercy_syst#A         0001110         0000000         1100101         0000000         00001110           Mowlatercy_syst#A         0001110         00000000         10100101         00000110	Messages							
1         Adoukatency_syn/R         0111101         1000010         1010010         1010010         0111101         0011001           1         Adoukatency_syn/R         0000000         1110101         0100110         1010010         00011001         0101110         00011001         0101100         00011001         0100110         0100110         0100110         0100110         0100110         0100110         0100110         0100110         0100110         0100110         010000         0000000         011000         0000000         011000         0000000         011000         0000000         011000         0000000         011000         0000000         011000         011000         011000         0000000         0100110         0000000         0100010         010000         011000         001100         0000000         0100010         0000000         001100         0001100         0000000         001100         0000000         0010100         0000100         0000110         0000000         0001100         0000000         0001100         0000010         0000110         0000100         0000110         0000110         0000110         0000110         0000110         0000110         0000110         0000110         0000110         00001100         0000110         000011	Mowlatency_sys/A 1100001	1110101	0101010	11100110	1110000	11100001		
Chowlatency_syst/M         001001         0100010         110001         000000         111000         0000000         0           Chowlatency_syst/M         000000         111000         000000         111000         0000000         0           Chowlatency_syst/M         000000         111000         000000         111000         0000000         0           Chowlatency_syst/M         000000         111000         100001         0000000         0		1000010	1010101	1000010	11110010		0111101	
1         Moviatercy_synM1         0000000         110000         0000000         0           1         Moviatercy_synM3         011000         0000000         110000         0000000           1         Moviatercy_synM3         0011100         0000000         0         0         0           1         Moviatercy_synM3         0011100         0000000         1100010         0000000         0	Mowlatency_sys/Y 001100:	0100010	1100110	11111101	0010111	0101110	0011001	
c.         Advalater(v_synAN2 Novlater(v_synAN2 C)         1110000 (000000         0000000 (110011         0000000 (100110         0110001 (100011         0000000 (100110         0000000 (100110         0000000 (100110         0000000 (100110         0000000 (100110         0000000 (100010         0000000 (100010         0000000 (100010         0000000 (100010         0000000 (100010         0000000 (100000         000000         0000000 (100000         0000000	Mowlatency_sys/M1 0000000	1110101	0101010	11100110	1110000	1100001	0000000	
novlatency_synAtt         0000000         110101         2010101         110010         0000000         00000110         0000000         00000110         0000000         0000110         0000000         00001110         0000000         00001110         0000000         00001110         00000110         000001110         0000000         00001110         000001110         0000000         00001110         000001110         000001110         0000000         00001110         000001110         000001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         00001110         000001110         00000000 <t< td=""><td>Mowlatency_sys/M2 1110000</td><td>0000000</td><td></td><td></td><td>0111000</td><td>11110000</td><td></td><td></td></t<>	Mowlatency_sys/M2 1110000	0000000			0111000	11110000		
1         Advokatercy_synAM4         0111000         0000000         1001010         0111000         0111000           1         Advokatercy_synAM4         0111000         0110000         0110000         0111000         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0011100         0001100         0001100         0001100         0001100         0000110         0000000         00101101         0000000         0001110         0000000         0001110         0000000         0001110         0000000         0001110         0000000         0001110         0000000         0001110         0000000         0001110         0000000         0001110         0000000         0001110         0000000         0001110         0000110	Mowlatency_sys/M3 0000000	A110101	0101010	11100110	1110000	11100001	0000000	
1         Movidatercy_symM5         1110000         1100000         1100000         1100000         1100000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         1000000         10000000         1000000         1000000         1000000         1000000         10000000         10000000         1000000         1000000		0000000	1001010	0000000	0011100	0111000		
Image: Movilatercy_sym/M6         0011100         0000000         110010         1001000         1000100         1000100         1000100         1000100         1000100         1000000         1001000         1000000         1000000         10000011         0000000         10000011         0000000         10000011         0000000         10000011         00000110         00000110         00000110         00000110         00000110         00000110         00000110         00000110         00000110         00000110         00000110         00000110         00000110         00000110         0000000         1001010         1000001         1001000         1000001         1001000         1000001         1001000         1000001         1001000         0000000         1001000         0000000         0000000         1001000         0000000         0000000         0000000         0000000         0000000         1000010	Mowfatency_sys/M5 1110000	1110101	10 10 10 10	1100110	1001000	0010001	1110000	
0-         //doubleterry_syn/M2         1001000         110101         1100000         1100100         1001000         1001000         1001000         1001000         1001000         1001000         1001000         1001000         1001000         1001000         1001000         1001000         1000100         1000100         1000010         1000000         1001100         1000000         1000011         1000000         1000011         1000000         1000011         1000000         1000011         1000000         1000011         1000000         1000011         1000000         1000011         1000000         1000011         1000001         100001         1000001	Mowlatency_sys/M6 0011100	0000000				_	0011100	
1         Mowlatercy_spiRN1         0001110         0000000         1010016         0000000         10001116           0         Mowlatercy_spiRN1         0000110         0000000         1010016         0000000         100001116           0         Mowlatercy_spiRN1         0001110         0000000         1010101         0000000         1000011           0         Mowlatercy_spiRN1         0001110         0000000         1010101         0000000         1000011           0         Mowlatercy_spiRN1         0001110         0000000         1010101         0000110         00001110           0         Mowlatercy_spiRN1         0001110         1010101         1010101         0001110         00001110           0         Mowlatercy_spiRN1         0001110         1010101         1010101         1000011         0000110         0000110           0         Mowlatercy_spiRN1         0001000         1100010         1100010         1100001         1000001         1000010         100001         100001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         1000001         10000		1110101	1100000	11100110	1010100	0101001	1001000	
	Mowlatency_sys/N1 0001110	0000000	10 100 10	0000000			0001110	
C         Movilatency_spiRAD         0001110         0000000         1010010         0000000         1000011           C         Movilatency_spiRAD         00001110         0000000         1010010         0000000         10000111           C         Movilatency_spiRAD         00001110         0000011         0000000         1010010         0000000         10000111           C         Movilatency_spiRAD         0001110         0101010         1010010         1000000         1100000         1100000         1100000         1100000         1100000         1000001         1000001         1000001         1000001         1000000         1100000         1000000         1000000         1000000         1000000         1000000         110000         10000000         10000000         1000000	All Mowletency_aya/N2 000000	1010111	0000000	0011011	1000011	0000111	0000000	
D         Mondatency_privAt         10000111         0000000         1010100         00000011         10000111           D         Mondatency_privAt         0000110         1010101         1010101         10100011         1000011           D         Mondatency_privAt         0001110         1010101         1010101         1010101         1010101         1010101           D         Mondatency_privAt         1010101         1010101         1010101         1010101         1010101         1010101           D         Mondatency_privAt         1000011         1010101         1010101         1010101         1010101         1010101           D         Mondatency_privAt         Sto         0         1010010         1110001         1010001         1000001         00000000           D         1000011         1110101         0101010         1100010         1100001         00000000         0000000 <th0< td=""><td>Mowlatency_sys/N3 0001110</td><td>0000000</td><td>110 100 10</td><td>0000000</td><td></td><td></td><td>0001110</td><td></td></th0<>	Mowlatency_sys/N3 0001110	0000000	110 100 10	0000000			0001110	
D-         Moviatercy_spiN5         0001110         101011         101001         0001101         1000011         0000110           D-         Moviatercy_spiN7         100100         110101         101000         101010         101010         101010           D-         Moviatercy_spiN7         100110         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         101010         100001         100001         1000000         100000         100000	Mowlatency_sys/N4 1000011	0000000	1010100	0000000			1000011	
n         Movidatency_syn(D1)         1010100         111010         1110000         1100110         1010100         1000101         1000101         1000101         1000101         1010000         1000001         00000000         1000001         1100000         1000001	Mowlatency_sys/N5 0001110	1010111	1010010	0011011	1000011	0000111	0001110	
no.         Movidatercy_grap.         100101         101011         1000001         1000001         1000001	Mowlatency_sys/01 1010100	1110101	1100000	1100110	1010100	0 10 100 1	1010100	
B-       Mowlatercy_systP       1100001       1110101       D010101       1100010       1100001       1000001         B-       Mowlatercy_systP       Sti       D000000       1110101       D010101       1110000       1100001       D000000         B-       Mowlatercy_systP       1100001       1110101       D010101       1110010       D000000       D0000000       D0000000       D00	Mowlatency_sys/02 100110:	1010111	10000110	0011011	1000011	0000111	1001101	
Movidations, grappin.         500           District, Statemark, grappin.         500           District, Statemark, grappin.         110001           District, Statemark, grappin.         110001           District, Statemark, grappin.         110001           District, Statemark, grappin.         110001           District, Statemark, grappin.         1100001           District, Statemark, grappin.         1100001           District, Statemark, grappin.         1100001           Movidatenark, grappin.         500           District, Statemark, grappin.         5000000           District, Statemark, grappin.         5000000           District, Statemark, grappin.         50000000           District, Statemark, grappin.         50000000           District, Statemark, grappin.         5000000           District, Statemark, grappin.         5000000           District, Statemark, grappin.         5000000           Distrit, Statemark, grappin.         50000000	Mowlatency_sys/P 1100001	1110101	0101010	1100110	1110000	1100001		
D-         //doubletercy_systP         0000000         1110101         1010101         1100010         1100000         00000000           D-         //doubletercy_systP         1100001         1110101         1010101         1100001         1100000         00000000           D-         //doubletercy_systP         1100001         1110101         10101010         1100001         1100001         0000000           D-         //doubletercy_systP         1100001         1110101         10101010         1110000         1100001         0000000           D-         //doubletercy_systP         1100001         1110101         1100101         1110000         1100001         0000000           D-         //doubletercy_systP         1100001         1110101         1100001         1100001         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         00000000         0000000         00000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         00000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         00000000         00000000         0000000 </td <td>/Nowlatency_sys/P St0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	/Nowlatency_sys/P St0							
Cl / Mondatency_grap.P         1100001         1110101         1010101         1100011         1100001           Cl / Mondatency_grap.P         1000001         1110101         0101010         1100001         1100001           Cl / Mondatency_grap.P         100001         1110101         0101010         1100001         1100001           Cl / Mondatency_grap.P         500         100001         1110101         0101010         1100001         1000001           Cl / Mondatency_grap.P         500         1000001         1110101         0101010         1100001         1000001           Cl / Mondatency_grap.P         500         0000000         110001         0101010         1100001         0000000           Cl / Mondatency_grap.P         1000001         1110101         0101010         1100001         0000000           Cl / Mondatency_grap.P         1000001         1110101         0101010         1100000         0000000           Cl / Mondatency_grap.P         1100001         1110010         0101010         11100000         0000000           Cl / Mondatency_grap.P         511         0000000         1110010         0101010         11100000         0000000           Cl / Mondatency_grap.P	Mowlatency_sys/P 0000000	1110101	0 10 10 10	1100110	1110000	1100001	0000000	
D - / Movilatency_systP         1100001         111010.         1010101         1100011         1100000           - / Movilatency_systP         1000001         1110101.         10101010         1100000         1100000           - / Movilatency_systP         550         0         1100001         1110000         1100000           - / Movilatency_systP         550         0         1100010         11100000         1100000           - / Movilatency_systP         1000001         111010         10101010         11100000         1000000           - / Movilatency_systP         1000001         111010         10101010         11100000         1000001           - / Movilatency_systP         1000001         1110101         10101010         11100000         1000001           - / Movilatency_systP         1100001         1110101         10101010         11100000         0000000           - / Movilatency_systP         1100001         1110010         11100010         0000000         0000000           - / Movilatency_systP         1100001         11100010         11100001         0000000         00000000         00000000           - / Movilatency_systP         1100001         00000000         01101010         11100010         <	Mowlatency_sys/P 1100003	1110101	10101010	1100110	1110000	1100001		
□-         //Movilatency_systP         1100001         1110101         0101010         1100010         1100001           □-         //Movilatency_systP         100001         1110101         0101010         1100010         1100001           □-         //Movilatency_systP         1000000         1110101         0101010         1100010         1100000           □-         //Movilatency_systP         1000000         1110101         0101010         1100010         1000000           □-         //Movilatency_systP         1000000         1110101         0101010         1100010         0000000           □-         //Movilatency_systP         1100000         0000000         1110101         0101010         1100010         0000000           □-         //Movilatency_systP         1100000         0000000         1110010         00000000         1100000         00000000           □-         //Movilatency_systP         1110000         00000000         1110010         01100001         00000000           □-         //Movilatency_systP         1110000         00000000         1110000         00000000	Mowlatency_sys/P 1100001	1110101	10 10 10 10	1100110	1110000	1100001		
Movidatency_gryuP         550           Display         1100001         111010         101010         1100001           Display         Movidatency_gryuP         0000000         1100011         1100001         1000001           Display         Movidatency_gryuP         0000000         1110101         0101010         1110001         1000001           Display         Movidatency_gryuP         1100001         1110101         1110010         1100000           Display         Movidatency_gryuP         1100001         1110101         1110010         1100000           Movidatency_gryuP         1100001         1110101         1110010         1100000         0000000           Movidatency_gryuP         Stit         0101010         1110010         1110000         0000000           Movidatency_gryuP         Stit         0101010         1110010         0000000         0000000           Movidatency_gryuP         Stit         0101010         11100010         0110000         0000000           Display         Stit         0110000         0000000         01110000         1110000           Stit         0110000         00000000         01110000         01110000         0110000	/Viowlatency_sys/P 110000:	11110101	10 10 10 10	1100110	1110000	11100001		
D- √ Mondatency_psyRP         1100001         111010.         1010101         1100001         100000.           D- √ Mondatency_psyRP         0000000         1110101.         0101010         11100001         0000000         1000001           D- √ Mondatency_psyRP         0000000         1110101.         0101010         11100001         0000000         1000001           D- √ Mondatency_psyRP         0000000         1110101.         0101010         1110000         1000001         0000000           D- √ Mondatency_psyRP         0000000         1110101.         0101010         1110000         1000001         0000000           D- √ Mondatency_psyRP         00000000         1110101.         0101010         1110000         1000001         0000000           III0000         0000000         1110101.         0101010         1100000         1000001         0000000           III0000         0000000         1110101.         0101010         1110000         1000001         0000000           III0000         00000000         1110010         01100001         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         00000000         0000000         00000	/Nowlatency_sys/P St0							
D-√         /Movilatency_systP         00000000         1110101         0101010         110010         0100000           D-√         /Movilatency_systP         0000000         1110101         0101010         1100010         0100001           D-√         /Movilatency_systP         0000000         1110101         0101010         1100001         0000000           Advalatency_systP         tst         010001         010000         0110000         0110000         0110000           C+         /Movilatency_systP         tst         0110000         0110000         0110000         0110000           C+         /Movilatency_systP         tst         0110000         0110000         0110000         0110000           C+         /Movilatency_systP         tst         0110000         0110000         0110000         0110000         0110000           C+         /Movilatency_systP         110000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         0000000         00000000         0000000         000	Mowlatency_sys/P 110000:	1110101	10 10 10 10	11100110	1110000	1100001		
n < //doubletercy_systP	Mowlatency_sys/P 0000000	1110101	10 10 10 10	1100110	1110000	11100001	10000000	
D-         /Mowlatency_systP         00000000         1110101         0101010         1100001         0000000           D-         /Mowlatency_systP         110000         0000000         01110000         1100000           D-         /Mowlatency_systP         110000         0000000         01110000         1110000	Mowlatency_sys/P 110000:	11110101	10 10 10 10	11100110	1110000	110000		
Anoulatercy_septP St      Comparison of the second secon	Mowlatency_sys/P 000000	(11)010	0 10 10 10	1100110	1110000	1100001	0000000	
D-         //doubtency_sys/P         1110000         0000000           2 #5 #         Now         1700 rs         1700 rs	Mowlatency_sys/P St1							
2 \$5 9 Nov 1700 ns	Mowlatency_sys/P 1110000	0000000			0111000	1110000		
	A Real Now	1700 ns						
Over 1 1568 m	Ourser 1	1568 ns					1665	(ne)

Fig. 16: low latency systolic structure

Messages						
	1110101	1100110	1100001	1110101		
M_TopModule/B	1000010	1000010	0111101	1000010		
	0100010	1111101	0011001	0 1000 10		
M_TopModule/M1	1110101	1100110	2000000	11110101		
M_TopModule/M2	0000000	0000000	20001110	0000000		
M_TopModule/A1	1110101	1100110	2000000	11110101		
M_TopModule/A2	0000000	0000000	Ž1110000	0000000		
M_TopModule/A3	1110101	1100110	1100001	11110101		
M_TopModule/A4	1010111	0011011	2000000	11010111		
M_TopModule/A5	0000000	0000000	0001110	0000000		
M_TopModule/B1	1110101	1100110	1110000	11110101		
M_TopModule/B2	0000000	0000000	0111000	0000000		
Now Now	900 ns	111111111111	200 ns 400 ns	600 ns	800 ns	100
Cursor 1	0 ms	0 ns				

## Fig. 17: Register Sharing Multiplier

Table 1: Device Utilization Summary Systolic

Device Utilization Summary					
Logic Utilization	Used	Available	Utilization	Note(s)	
Number of 4 input LUTs	420	7,168	5%		
Logic Distribution					
Number of occupied Slices	210	3,584	5%		
Number of Slices containing only related logic	210	210	100%		
Number of Slices containing unrelated logic	0	210	20%		
Total Number of 4 input LUTs	420	7,168	5%		
Number of bonded IOBs	63	141	44%		
Total equivalent gate count for design	3.087				
Additional JTAG gate count for IOBs	3.024				

Table 2: Low Latency Systolic Architecture

Device Utilization Summary					
Logic Utilization	Used	Available	Utilization	Note(s)	
Number of 4 input LUTs	399	7,168	5%		
Logic Distribution					
Number of occupied Slices	200	3,584	5%		
Number of Slices containing only related logic	200	200	100%		
Number of Slices containing unrelated logic	0	200	0%		
Total Number of 4 input LUTs	399	7,168	5%		
Number of bonded IOBs	63	141	44%		
Total equivalent gate count for design	2,835				
Additional JTAG gate count for IOBs	3.024				

Table 3: Register Sharing Lo	w Latency Systolic Architecture
------------------------------	---------------------------------

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	399	7,168	5%	
Logic Distribution				
Number of occupied Slices	200	3,584	5%	
Number of Slices containing only related logic	200	200	100%	
Number of Slices containing unrelated logic	0	200	0%	
Total Number of 4 input LUTs	399	7,168	5%	
Number of bonded <u>IOBs</u>	63	141	44%	
Total equivalent gate count for design	2,772			
Additional JTAG gate count for IOBs	3,024			

#### CONCLUSION

Efficient systolic design for the multiplication over GF (2<sup>m</sup>) based on irreducible AOP is proposed. This derived a low-latency bit-parallel systolic multiplier. Compared with the existing systolic structures for bit-parallel realization of multiplication over GF (2<sup>m</sup>), the proposed one is found to involve less area, shorter critical-path and lower latency. From ASIC and FPGA synthesis results to find that the proposed design involves significantly less ADP and PDP than the existing designs. Moreover, the proposed design can be extended to further reduce the latency. The usage of Noval Cut-set Retiming reduces the critical path to one XOR gate thus the complex systolic structure have been spiltted into two or more parallel systolic branches and each one is fed with same input operand and shares same input operand register thus by using irreducible All One Pollynomialalgorithm, overall circuit complexity is reduced and which can be implemented for cryptography and error control technique.

### REFERENCES

- Berk Suran, 2004. A Generalized Method for constructing subquadratic complexity GF(2<sup>k</sup>) Multipliers, IEEE Trans. Computers, 53(9): 1097-1105.
- Hanho Lee, 2003. High speed VLSI architecture for parallel Reed-Solomon decoder, IEEE Trans. Coputers, 11(2): 288-294.
- Neethu Johny and Binoy Joseph, 2013. An efficient Systolic multiplier forGF(2m) based on All One Polynomial,journal, Trans. Computers, 1: 2320-2351.

- Chiou Yng Lee and Che Wun Choiu, 2005. Low complexity bit parallel dual basis multipliers using the modified Booth's algorithm, journal Trans.computers, 31: 444-459
- Jean Claude, 2010. Subquadratic Space complexity Binary field multiplier using double polynomial representation, IEEE. Trans. Computers, 59(12): 1585-1597.
- Henriqez, 2003. Parallel Multipliers Based on Special Irreducible pentanomials, IEEE, Trans. Computers, 52(11): 1-7.
- Yadollah Eslami, 2006. An area efficient Universal cryptography processor for smart cards, IEEE, Trans. Computers, 14(1): 44-51
- Leong, H.W., 2002. Amicrocoded elliptic curve processor using FPGA technology, IEEE. Trans. Computers, 10(5): 550-559.
- Cancio Monterio, 2013. Low power bit parallel cellular multiplier implementation in secure dual rail adiabatic logic, IEEE. Trans. Computers, 3(4): 10-19.
- Bimal Kumar Meher, 2009. A effectiveness of various implementation options of finite field arithmetic on elliptic curve cryptosystem, IEEE. Trans. Computers, 3(4): 1793-8201.
- Ashutosh Kumar singh, 2009. Error detecting dual basis bit parallel systolic multiplicationar chitectureover GF(2m), Jornals. Trans. Computers, 7(4): 336-342.
- Kazutoshi Wakabayashi, 2000. C-Based SoC design flow and EDA tools an ASIC andsystemVendor prespetive, IEEE. Trans. Computers, 3(4): 1507-1522.