# MODIFIED TIME MULTIPLEXED SYSTOLIC ARRAY FOR FINITE FIELD MULTIPLICATION OVER GF ($2^m$) BASED ON IRREDUCIBLE ALL-ONE POLYNOMIALS

Duggirala Sowjanya[1], Dr. Pushpa Kotipalli[2]
[1]Shri Vishnu College for Women. Vishnupur, Bhimavaram, West godavari district, Andhra pradesh,
[2]Shri Vishnu College for Women. Vishnupur, Bhimavaram, West godavari district, Andhra pradesh,

***Abstract :*** In this paper, an efficient finite field multiplication is suggested over GF($2^m$) based on irreducible AOP. The Redundant basis (RB) multipliers over Galois Field (GF ($2^m$)) have gained huge popularity in elliptic curve cryptography (ECC) mainly because of their negligible hardware cost for squaring and modular reduction. In this paper, The bit-parallel beat structure has an equivalent cycle time as that of the most effective existing bit-parallel systolic structure [1], however involves considerably less range of registers. The bit-parallel design has a scalable latency of *l* + _log2 *s*_ + 1 cycles which is considerably low compared with those of existing systolic designs. Moreover, the proposed time-multiplexed structure is designed specifically for scalability of throughput and hardware-complexity to meet the area-time trade-off in resource-constrained applications while maintaining or reducing the overall latency. Both theoretical analysis and synthesis results con firm the efficiency of proposed multiplier over the existing ones. The synthesis results for field programmable gate array (FPGA) and application specific integrated circuit (ASIC) realization of the planned styles and competitive existing designs are compared. It's shown that the proposed high-throughput and low latency structure is the best among the corresponding designs, for FPGA and ASIC implementation.

***Index Terms*** **- Galois Field(GF), All-One Polynomial(AOP), Elliptic Curve Cryptography (ECC).**

## I.INTRODUCTION

Cryptography is that the sciences of hiding data which may be disclosed solely by legitimate users. It is used to ensure the secrecy of the transmitted data over an unsecure channel and prevent eavesdropping and data tampering [1]. Many cryptography schemes were proposed and used for securing data, some uses the shared key cryptography and some uses the public key cryptography (PKC). Shared key cryptography could be a system that's uses just one key by each sender and receiver for purpose of encrypting and decrypting the message. On the other hand, public key cryptography uses two keys, private-key and public-key. To encrypt a message in Public key scheme, public-key will be used and to decrypt it back a private-key is used.

As compared to the shared key cryptography, public key cryptography is slow. However, public-key cryptography can be used with shared key cryptography to get the best of both. Public key cryptography have many advantages over the shared key, it increases the security and convenience where there is no need to distribute the private key to anyone [2]. Most of today's application of cryptography asks for authentication and secrecy of the data. Secret transmission of data is an important task to preserve the data from the immune to attacks, threats and misuse. The encrypted text or data is less secure since it can be easily decrypted. But an image cannot be easily decrypted by attackers. Even data can be transmitted more securely by converting it into an image.

The most of hardware and software products and standards that use public key technique for encryption and decryption, authentication etc. are based on RSA cryptosystem by using non Conventional algorithms among RSA and ECC. The main attraction of ECC is that it can provide better performance and security for small key size, in comparison of RSA cryptosystem. ECC is not easy to understand by attackers. So it provides better security through insecure channels.

In 1985, Neal Koblitz and Victor Miller independently planned public key cryptosystems exploitation elliptic curve. Since then, many researchers have spent years studying the strength of ECC and improving techniques for its implementation. Elliptic curve cryptosystem (ECC) provides a smaller and faster public key cryptosystem. ECC has been commercially accepted, and has conjointly been adopted by several standardizing bodies like ANSI, IEEE, ISO and government agency. The operation of each of the public-key cryptographic schemes described in this document involves arithmetic operations on an elliptic curve over a finite field determined by some elliptic curve domain parameters.

The main attraction of ECC is that it can provide better performance and security for small key size, in comparison of RSA cryptosystem. In ECC a 160-bit key provides the same security as compared to the traditional crypto system RSA with a 1024-bit key, thus in this way it can reduce computational cost or processing cost. The security of ECC depends on the difficulty of finding the multiplicand for the given product and multiplier. ECC is not easy to understand by attackers. So provides better security through insecure channels.

## II.LITERATURE REVIEW

Elliptic curves have been studied for over hundred years and have been used to solve a diverse range of problems. The use of elliptic curves in public key cryptography was first proposed independently by Koblitz and Miller. ECC generates keys through the properties of the elliptic curve equation instead of the standard methodology of generation because the product of very large prime numbers. The technology can be employed in conjunction with most public key cryptography ways, like RSA, and Diffie-Hellman [3].

The main advantage of ECC over conventional asymmetric crypto systems [4] is the increased security offered with smaller key sizes. For example, a 256 bit key in ECC produces the same level of security as a 3072 bit RSA key1. The smaller key sizes leads to compact implementations and increased performance. The scheme for implementation of finite field multiplication over GF($2^m$) generated by AOP can be outlined as shown in Fig. 1.

The finite field multiplication of A and B over GF($2^m$) is given by

$$C = A .B \bmod Q (Z), \quad (1)$$

The algorithm for multiplication explained as follows:

- STEP-1: Perform multiplication of bit b0 with the input operand A, to obtain b0 • A and initialize the first (m−1)-bits of a finite field accumulator by (b0.ai ), for $0 \leq i \leq m - 1$ . The m-th location (i.e., the MSB) of the FFA is initialized to zero.
- STEP-2: For i = 1 to m − 1
- Perform cyclic left-shift operation of the polynomial Pi−2α of degree (m + 1) to reduce its degree by one to obtain the operand Pi−1 of degree m.
- Perform bit-level multiplication of bi with Pi−1 to obtain Yi
- Add Yi to the content of the FFA to obtain the partial result of degree m
- STEP-3: Perform modular reduction of Y to reduce its degree from m to (m − 1) to obtain the desired product value.

Note that all the recursive operations of the algorithm are in STEP-2, while STEP-1 may be considered as pre-processing step and STEP-3 may be considered as a post-processing step.

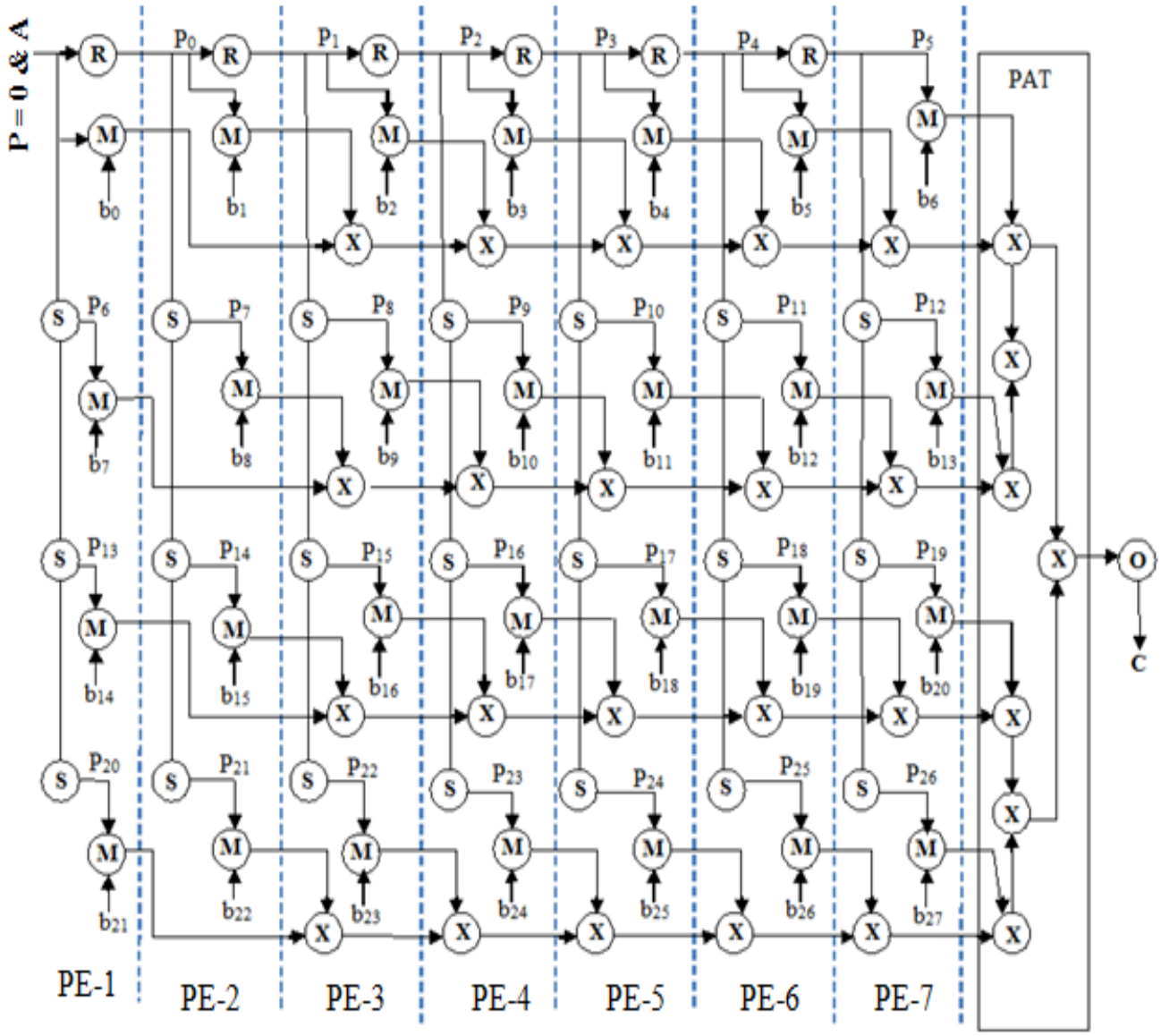


Fig. 1. The dependence graph for the finite field multiplication over GF ($2^m$) for irreducible AOP

For systolic implementation of multiplication over GF ($2^m$), the operations of STEP-2 can be performed recursively, where each recursion is comprised of three steps, e.g., the modular reduction by one, bit-multiplication operations followed by the field additions [5]. This architecture consists of (m − 2) modular reduction nodes "R," (m − 1) field addition nodes "X" and equal number of bit-multiplication nodes "M." One more bit-multiplication node is required for pre-processing of STEP-1 and an output reduction node "O" is required for the post-processing of STEP-3. The function of the reduction nodes is illustrated in Fig. 1. It performs the modular reduction of degree by one The DG could be retimed by the cut-sets shown by dashed lines in Fig. 1 in to m logic units (LU), and a pair of post-processing units

**III. METHODOLOGY**

In two-dimensional parallel systolic array, where each row contains *l* LUs. The value of *l* is related to the latency of the multiplication and should be determined by the requirement of the application [6]. Therefore, the number of rows in the array is s = m/*l*. The last row might have less number of LUs if m is not integer multiples of *l*. Therefore, for a finite field of order m, we generally have

$$m = ls - r, \quad (2)$$

Where r is an integer in the range of [0, *l*]. For the cases of r > 0, the multiplicands can be padded with r -bit zeros [7]. The construction of a two-dimensional DG for m = 28 is shown in Fig. 2 as an example, where we choose l = 7, s = 4, and r = 0 for the convenience of explanation. It consists of four rows, where each row consists of five regular LUs and two pre-processing LUs. Each LU (except the top row) requires a multi-reduction cell S, to perform the reduction of degree by l to generate appropriate operands for the next row of LUs. In the same as node R, the multi-reduction node S can be realized by l-bit cyclic left shift, which can be implemented by wire mapping. To have the regularity of structure of different rows, we can take the input P = 0&A obtained by appending a 0 with the input operand A, such that after one cyclic left-shift operation it would generate the operand P = $P_0$. It can be partitioned by the cut-sets shown by vertical dashed lines in the Fig. 1.

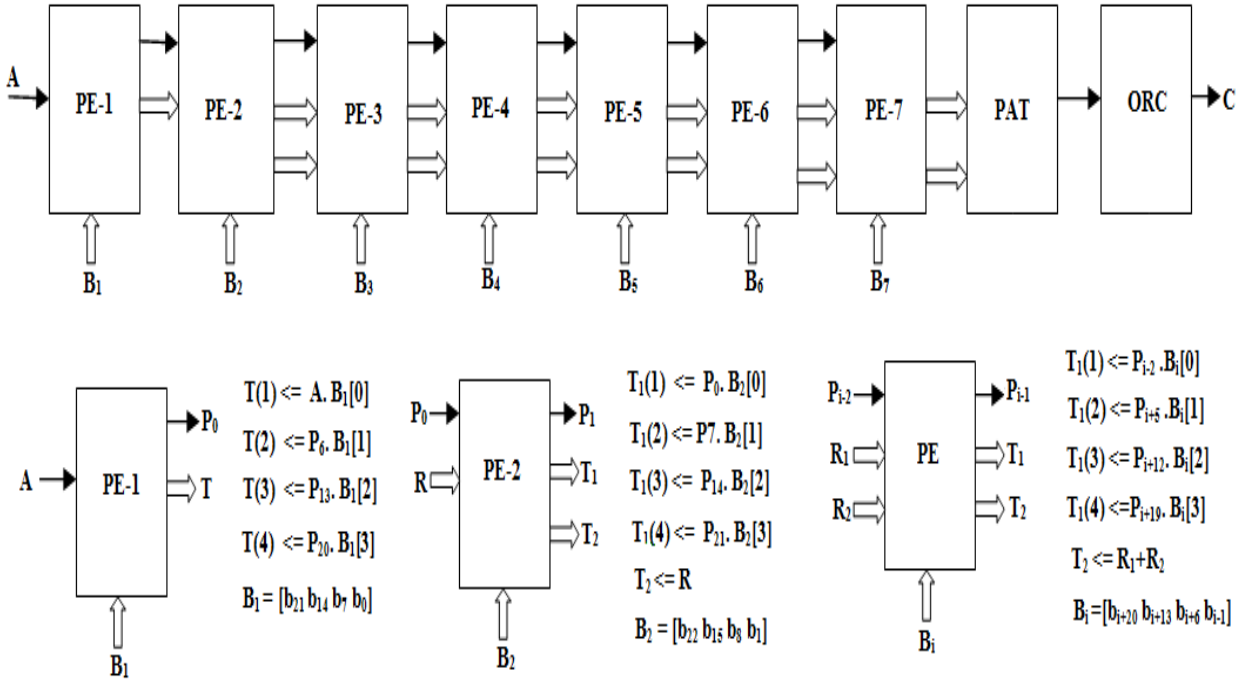


Fig. 2. Parallel systolic-like array for the finite field multiplication over GF($2^m$) based on irreducible AOP. (a) The linear array structure. (b) Function of PE-1. (c) Function of PE-2. (d) Function of the ith regular PEs (PE-3 to PE-7).

The functions of LUs of each column are embedded in the respective PE. A pipelined-adder-tree (PAT) is used to implement the post-processing logic followed by an Output Reduction Cell (ORC). Each of the PEs take 4 bits of input B in every cycle. The functions of different types of PEs are shown in Figs. 2(b), 2(c), and 2(d). Each of the first PE and the second PE (PE-1 and PE-2) consists of four AND cells to perform the function of 4 multiplication nodes M. Each of the other PEs (the regular PEs: PE-3 to PE-7) consists of four AND cells and four XOR cells to perform the functions of multiplication nodes and Addition nodes. Except the last PE, all other PEs requires one reduction cell to implement the function of node R. The reduction nodes are implemented by rewiring of bits, which do not require combinational resources. The last PE or PE-7 does not require the operation of reduction node R, otherwise its function is the same as other regular PEs. Total latency of this structure is 10 cycles (7 cycles in PEs, 2 cycles in PAT and 1 cycle for ORC), and it produces one product word in each cycle once the pipeline is filled-in during the latency period. The duration of cycle period $T = T_X$, where $T_X$ is the delay of an XOR gate. One of the primary advantages of the parallel systolic architecture over other systolic architectures is that we can have alterable latencies in terms of cycles by choosing different *l* and s values while maintaining the cycle time and throughput. Generally, the latency of the architecture in Fig. 2 can be expressed as

$$L = l + [\log 2\ s] + 1, \quad (3)$$

Where *l* is the latency for the *l* PEs, [ $\log_2$ s] is the latency for the PAT and the constant 1 is for the ORC. There could be several other choices of *l* and s for partitioning the DG for a given value of m. For example, another choice for the partitioning of DG of Fig. 2 could be *l* = 5, s = 6, and r = 2. In this case, we shall have 5 PEs, but the pipelined-adder-tree will have one additional XOR cell and the last two PEs will have two redundant delays to wait for other PEs to complete their tasks. But the pipelined-adder-tree will have one additional XOR cell and the last two PE's will have two redundant delays to wait for other PEs to complete their tasks.

$$L = [m/s] + [\log_2 s] + 1, \quad (4)$$

In real practice, the value of l and s can be determined according to the latency requirement of the application. Parallel systolic – like array for the finite field multiplication over GF ($2^m$) based on irreducible All One Polynomial takes more delay than the expected result. In order to avoid this problem we derived a modified time multiplexed systolic like structure of multipliers for GF ($2^m$) based on irreducible AOP.

### *3.1 Modified Time -multiplexed systolic-like array (TM- 4) for the finite field multiplication over GF ($2^m$) based on irreducible AOP*

An efficient implementation of the rows of LUs of the 2-D DG in Fig. 1 can be time-multiplexed. The most straightforward and hardware efficient way is to design the hardware for four rows of Fig. 1and time-multiplex it. This is referred to as TM-4 structure, where 4 represent the time-multiplexed hardware for four rows. The TM-4 structure of multiplier for GF ($2^m$) based on irreducible AOP is shown in Fig. 3 for m = 28. It consists of four PEs, pipelined-adder-tree (PAT) and one output reduction cell (ORC). The functions of different PEs are performed horizontally instead of vertically shown in Fig. 1. To have the regularity of structure of different rows, we can take the input P = A&0 obtained by appending a 0 with the input operand A. PE-1 consists of six reduction nodes 'R', seven AND cells and five XOR cells. While each of the remaining PEs (PE-2 to PE-4) requires seven multi-reduction nodes 'S', seven AND cells and six XOR cells. The reduction node can be realized by one bit cyclic left shift. The

reduction nodes are implemented by rewiring of bits, which do not require combinational resources. In the same as node R, the multi-reduction node S can be realized by 7-bits cyclic left shift, which can be implemented by wire mapping. It reduces the input operand by order 7 in each cycle iteratively to generate successive input operands for the systolic array. The pipeline-adder-tree (PAT) is used to implement the post-processing logic followed by an Output Reduction Cell (ORC). Each of PEs takes 7 bits of input operand B in every cycle. Each PE consists of seven AND cells to perform the function of 7 multiplication nodes ' M' and five XOR cells to perform the function of five addition nodes ' X'. There are two partial outputs generated by each PE. The outputs of each PE fed to the Pipelined-adder-tree (PAT). It consists of XOR cells to perform addition operation. Finally the contents of PAT are transferred to the ORC. It reduces the degree by one to generate the product word C. The latency of this structure is 6 cycles (4 cycles for the PEs for the first partial result, 1 cycle for the PAT and 1 cycle for the ORC).

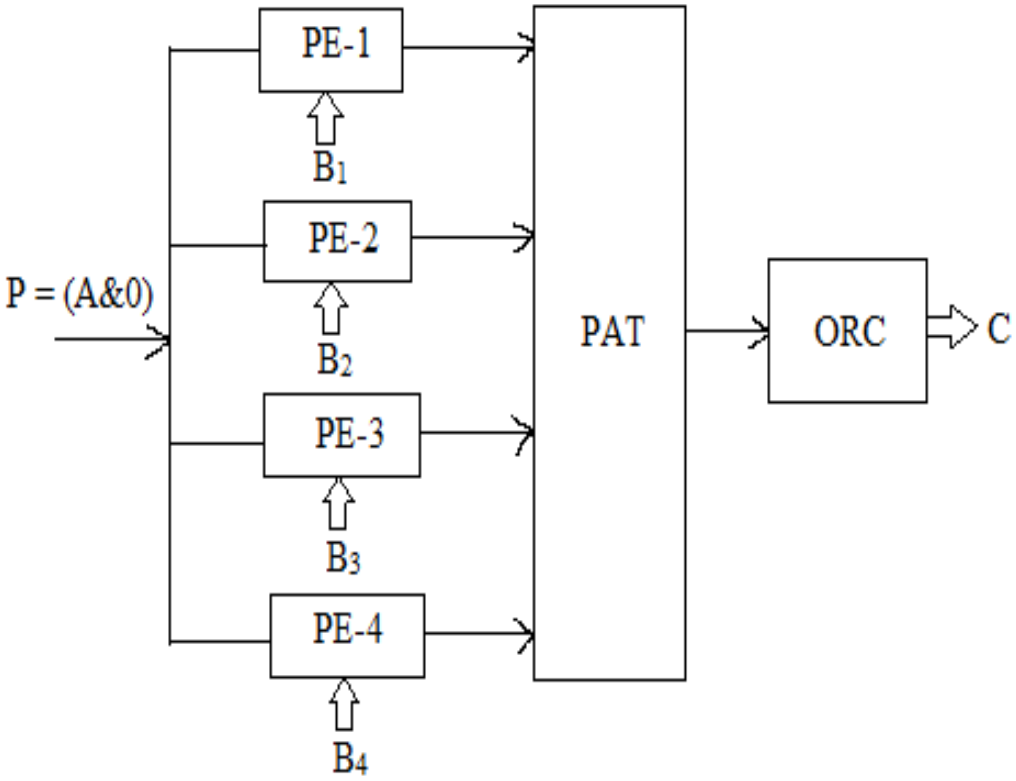


Fig. 3. The proposed Time-multiplexed systolic-like array (TM- 4) for the finite field multiplication over GF($2^m$) based on irreducible AOP

The TM-4 structure is improves the performance in terms of latency, and it has relatively high throughput than the parallel structure because the in parallel systolic structure in each PE need to wait for the other neighbour processing element results. The TM-4 structure of Fig. 3 involves same hardware and provides 4 times high throughput with an decrease in latency by 4 cycles compared with the structure in Fig. 2. Moreover, by designing the hardware for n rows and time multiplexing it, the TM-4 structure can be easily generalized to TM-n structure to trade-off the hardware complexity against latency and throughput. In the TM-n structure, the DG in Fig. 1 are partitioned into [s/n] sets of rows, where each set contains n rows (the last set may contain less than n rows). Each time a set is processed in parallel while all the sets of rows are processed in time-multiplexed to be processed one after the next. The general structure of TM-n is similar to that of TM-4. In TM-n, all the PEs need to process the computation of n rows. Moreover, a PAT with [$\log_2$ n] stages is needed to sum the results of n partial results. Compared with TM-4, the TM-n structure provides n times more throughput as well as decreased latency, while it consumes n times more hardware.

The area can be traded further for time in the time-multiplexed structure TM-n for 2 or higher, where the degree of folding is reduced. As can be seen, although both the field size and the cycle time of m = 100 are less than that of m = 148, the ACT of m = 100 is significantly larger than that of m = 148 because for m = 100 the choice of the s value is s = 10, while for m = 148, s = 4 is used. The reduction of latency could be more significant for higher order fields with large s values. it therefore produces the first output word after (l+s+1) cycles, and the successive output are produced by this structure in every s cycles. The actual computation time (ACT) for one product is only $sT_X$, where $T_X$ is the delay of a 2-input XOR gate. One of the interesting features of this design is that it has very low latency of (l + s + 1) cycles. Moreover, as discussed, the throughput rate of the time multiplexed design can be scaled by using the TM-n to multiplex the computation of n rows.

The throughput of the TM-n structure is n times higher than that of the TM-1 structure. Unlike conventional digit-serial structures, where both the hardware complexity and latency are n times larger, the TM-n structure can reduce the latency by s – n – [$\log_2$ n] when increasing the throughput to be n times higher. The area and time complexities of this design gives best result by comparing with the other existing bit/digit-serial systolic structures [8] for finite field multiplication based on All-One Polynomials (AOPs). There are several bit-serial designs and digit-serial designs. As can be seen, the bit-serial designs always

have the minimum hardware complexity, but also the minimum throughput rate. The digit-serial designs have moderate hardware complexity as well as throughput in terms of clock cycles. Therefore, the Average Computation Time (ACT) of the proposed TM-4 and TM-n designs are significantly smaller than that of Talapatra_DS [9]. Moreover, the throughput of TM-n can be easily scaled without sacrificing the overall latency of the structure.

## IV. RESULTS AND DISCUSSION

4.1 Results of Descriptive Statics of Study Variables

| Parameter | Parallel systolic array | Time multiplexed systolic array |
|---|---|---|
| Area($mm^{2)}$ | 15402 | 15402 |
| Critical Path Delay(ns) | 0.958 | 0.742 |
| Power(mw) | 2.875 | 2.875 |
| Area Delay Product | 14755 | 11428 |
| Power Delay Product | 2.754 | 2.133 |

## V. CONCLUSION

In this project, an efficient recursive formulation is suggested for systolic implementation of canonical basis finite field multiplication over GF ($2^m$) based on irreducible AOP, where the necessary modular degree-reduction is achieved by cyclic-left-shift operations. Suitable DGs are derived from the recursive formulation and systolic multipliers are designed there from. In the time multiplexed systolic designs, the cyclic-left-shift operations for modular reduction is implemented by shift-registers, which also act as latches between the PEs in a pure systolic pipeline. It is interesting to note that the time multiplexed systolic structure does not require any global communication for modular reduction. The proposed time multiplexed systolic structure has a similar cycle time as that of the most effective existing bit-parallel systolic structure, however involves considerably less range of registers.. The proposed time multiplexed design has a scalable latency of which is considerably low compared with those of existing systolic designs. Moreover, the proposed time-multiplexed structure is designed specifically for scalability of throughput and hardware complexity to meet the area-time trade-off in resource constrained applications while maintaining or reducing the overall latency.

## VI.REFERENCES


[1]. Deyan Chen Hong Zhao, Data Security and Privacy Protection Issues in Cloud Computing, and ICCSEE, 2012 International Conference on (Volume: 1) ePrint23-25 March 2012the IEEE website. http://www.ieee.org/

[2]. Parsi Kalpana, Sudha Singaraju, Data Security in Cloud Computing using RSA Algorithm, International Journal of Research in Computer and Communication technology, IJRCCT, ISSN 2278-5841, Vol 1, Issue 4, September 2012.

[3]. Neha Tirthani, and Ganesan. R R, Data Security in Cloud Architecture Based on Diffie Hellman and Elliptical Curve Cryptography, International Association for Cryptologic Research Cryptology ePrint 4-9, 2014.

[4]. P. K. Meher, "Systolic and super-systolic multipliers for finite field GF(2m) based on irreducible trinomials," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 5, pp. 1031–1040, May 2008.

[5]. A. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over GF(2m)," IEEE Trans. Comput., vol. 53, no. 8, pp. 945–959, Aug. 2004.

[6]. F. Rodriguez-Henriguez and C. K. Koc, "Parallel multipliers based on special irreducible pentanomials," IEEE Trans. Comput., vol. 52, no. 12, pp. 1535–1542, Dec. 2003. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over GF($2^m$)," IEEE Trans. Comput., vol. 53, no. 8, pp. 945-959, Aug. 2004.

[7]. R. Katti and J. Brennan, "Low complexity multiplication in a finite field using ring representation," IEEE Trans. Comput., vol. 52, no. 4, pp. 418–427, Apr. 2003.

[8]. S. T. J. Fenn, M. G. Parker, M. Benaissa, and D. Taylor, "Bit-serial multiplication in GF ($2^m$) using irreducible all-one polynomials," IEEEproc, comput. Digit. Tech., vol. 144, no. 6, pp. 391-393,1997.

[9]. P. K. Meher, "High-throughput hardware-efficient digit-serial architecture for field multiplication over GF(2m)," in Proc. 6th Int. Conf. Inf. Commun. Signal Process. (ICICS), Dec. 2007, pp. 1–5.