ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

Vol.02, Issue.01
July -2020
Pages: -215-224

LOW DENSITY AND LOW COMPLEX MODULAR MULTIPLIERS FOR POLYNOMIALS USING RADIX 16 MODIFIED BOOTH ALGORITHM

[1] SHAIK GOUSIYA, [2] MD.SHAMSHAD BEGUM

[1]M.Tech, Dept. of ECE, Nimra College of Engineering and Technology, Ibrahimpatnam, Vijayawada, A.P

[2]Assistant Professor, Dept. of ECE, Nimra College of Engineering and Technology, Ibrahimpatnam, Vijayawada, A.P

ABSTRACT: By using irreducible AOP, an effective recursive formulation is proposed implies systolic implementation of these finite field multiplications over GF (2m). Here, a recursive algorithm derived for the multiplication and used it in designing a systematic and localized bit-linear dependence graph for computing systolic multiplication. This dependence graph is altered to a fine-grained dependence graph (DG) by using node splitting method. This parallel systolic architecture is mapped from fine grained DG. Compared to other structures, it doesn't include any global communication for reducing the modules. Further as an extension of this concept, Radi16 modified booth encoding algorithm is proposed to reduce parameters for implementation of multiplier architecture.

***INDEX TERMS**: dependence graph, Latency, Multiplier, parallel systolic, Modified booth encoding.*

INTRODUCITON: F INITE FIELD multiplication over Galois Field ( ) is a basic operation frequently encountered in modern cryptographic systems such as the elliptic curve cryptography (ECC) and error control coding [1]–[3]. Moreover, multiplication over a finite field can be used further to perform other field operations, e.g., division, exponentiation, and inversion [4]–[6]. Multiplication over can be implemented on a general purpose machine, but it is expensive to use a general purpose machine to implement cryptographic systems in cost-sensitive consumer products. Besides, a low-end microprocessor cannot meet the real-time requirement of different applications since word- length of these processors is too small compared with the order of typical finite fields used in cryptographic systems. Most of the real-time applications, therefore, need hardware implementation of finite field arithmetic operations for the benefits like low-cost and high-throughput rate. The choice of basis to represent field elements, namely the polynomial basis, normal basis, triangular basis and redundant basis (RB) has a major impact on the performance of the arithmetic circuits [7]–[9]. The multipliers based on RB [6], [10] have gained significant attention in recent years due to their several advantages. Not only do they offer free squaring, as normal basis does, but also involve lower computational complexity and can be implemented in highly regular computing structures [10]–[14]. There are different types of bases to represent field elements, those are polynomial basis, normal basis, triangular basis and RB, and the choice of representation of field elements has a major impact on the performance of the arithmetic circuits [7]–[9], [15]. Several algorithms for basic arithmetic operations in GF ($2m$) are suitable for both hardware and software implementations have been recently developed. Because of several advantages of the RB based multipliers [6], [10] they have gained significant attention in recent years. Like normal basis multipliers, RB multipliers offer free squaring, they also involve lower computational complexity and can be implemented in highly regular computing structures [10]–[14]. Several digit-level serial/parallel structures for RB multiplier over GF ($2m$) have been reported in the last few years [10]–[14]. An efficient serial/parallel multiplier using redundant representation has been presented [10].

LITERATURE SURVEY: RB representation has high modularity and exhibits carry-free addition,

which can be used to design high performance multipliers. The main idea of multiplication using redundant representation is to perform multiplication by embedding the field in a larger ring. The ring used here is a cyclotomic ring and has a very simple structure, such that the modular operation can be saved in a multiplication operation. The main drawback for redundant representation is that it uses more bits to represent an element as compared to other representation basis. The number of representation bits depends on the size of the cyclotomic ring. However, for the class of fields GF (2*m*) for which there exists a type I optimal normal bases (ONB), the number of bits required for a redundant representation of a field element is slightly higher for large m—(m+1) bits compared to m bits used for the other bases. . These multipliers possess common properties like regularity, locality and reclusiveness. Previous work was targeted on the literature survey conducted on various multipliers in VLSI. It included the study of the basic technical aspects behind the design approaches of proposed systolic array multiplier. Systolic array is defined as a connected set of processors with rhythmical data computation and propagation along the system. They result in cost effective high performance special purpose system for a wide range of problems. By using this systolic array, concurrency and communication is achieved and special purpose design cost can be reduced by the use of special architecture. But one must provide a convenient means for incorporating high performance systolic processors into a complete system [1]. The systolic array designs are optimized for speed. After studying these designs, it is concluded that for multiplication of large matrices memory architectures is quiet efficient than systolic array. It requires several clock cycles [2]. There are various systolic architectures. These gives solution for the addressed issues of demand for high speed data processing. Also the number of components required for the matrix multiplication in conventional method is more than that of required in systolic method. For example critical path delay in conventional method is 9.831ns and in systolic array, it is 4.757ns.This system enhance the speed and reduce the complexity. It requires less number of clock cycles. It requires more number of accumulator than conventional method [3].

MULTIPLIERS FOR *GF(2M)* BASED ON IRREDUCIBLE AOP:

For systolic implementation of multiplication over *GF(2m)*, the operations of STEP-2 given by (11) and (13) can be performed recursively, where each recursion is comprised of three steps, *e.g.*, the modular reduction of (11), bit-multiplication operations of (13c) or (13d) followed by the field additionsof (13b). The said recursions can be represented by the dependence graph (DG) shown in Fig. 1, which 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 preprocessing 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(b). It performs the modular reduction of degree by one according to (11). Function of the bit-multiplication nodes, Addition nodes and output reduction nodes are depicted in Figs. 1(c), 1(d), and 1(e), respectively. Each of the bit-multiplication nodes performs an

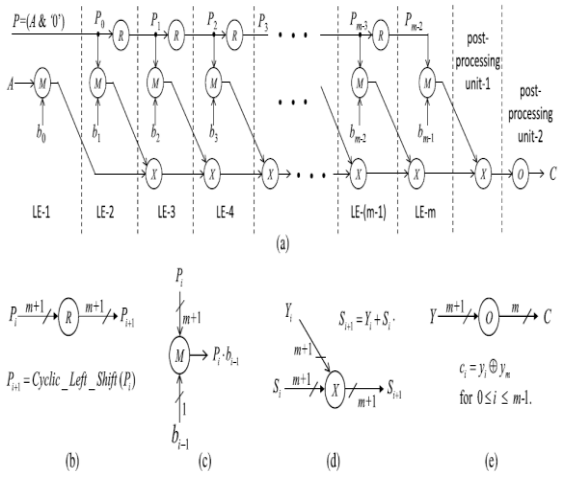


Fig. 1. The dependence graph (DG) of the recursive formulation of the finite field multiplication over *GF(2m)* based on irreducible AOP. (a) The DG. (b) Functional description of reduction node *R*. (c) Functional description of bit-multiplication node *M*. (d) Functional description of the Addition node *X*. (e) Functional description of the output reduction node *O*.

AND operation of a bit of operand *B* with a reduced form of operand $(\alpha^i \cdot P)$, which is polynomial of degree *m*, while each of Addition nodes performs an addition operation. The DG could be retimed by the cut-sets shown by dashed lines in Fig. 1(a) in to *m* logic units (LU), and a pair of postprocessing units. Based on this retiming, it is straight-forward to map this DG onto a linear

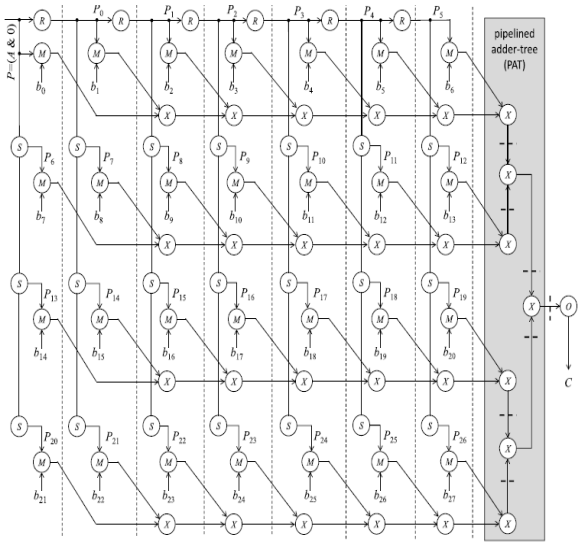
systolic array consisting of m processing elements (PE) along with two post-processing cells. The resulting linear systolic array (not shown in figures) can provide very high-throughput of one product word per cycle, where the cycle period is only one XOR delay. But, such a linear systolic design has two major disadvantages.

1. First, the latency of the structure is nearly m, where m is the field order.
2. Secondly, the register complexity of the structure far exceeds the complexity of combinational circuit, since each PE would have an AND gate and an XOR gate to implement the combinational logic but three registers to transfer bits to the neighboring PE.

To avoid these problems, we derive here a parallel structure of multipliers for *GF(2m)* based on irreducible AOP with low register complexity. Moreover, for hardware-efficient realization we have proposed a time-multiplexed structure, where throughput can be traded-off against area with moderate increase in latency.

PARALLEL SYSTOLIC STRUCTURE:

Instead of mapping the DG in Fig. 1 into a one dimensional linear systolic array, we propose to partition the one-dimensional DG by LUs and rearrange the LUs into a 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. Therefore, the number of rows in the array



a)

(b)

Fig. 2. The regularized dependence graph (DG) for the finite field multiplication over *GF(2m)* for irreducible AOP. (a) The DG. (b) Function description of multi-reduction node *S*

is $s = \lceil m/l \rceil$. 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 $m = ls - r$, (16) where r is an integer in the range of $[0, l]$. For the cases o $r > 0$, the multiplicands can be padded with r -bit zeros. 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 $P-1 = 0\&A$ obtained by appending a 0 with the input operand A, such that after one cyclic left-shift operation it would generate the operand $P = P0$. It can be partitioned by the cut-sets shown by vertical dashed lines in the figure, and mapped onto a linear systolic array of Fig. 3, where the functions of LUs of each column is embedded in the respective PE. A pipelined-addertree (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. 3(b), 3(c), and 3(d). Each of the first PE and the second PE (PE-1 and PE-2) consists of four AND cells to

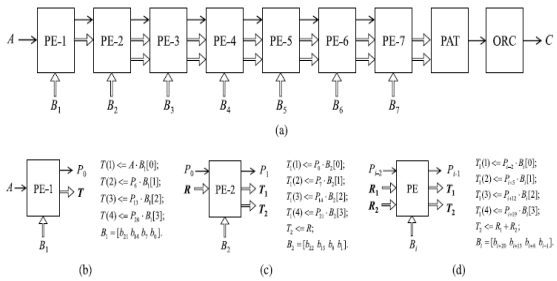


Fig. 3. The proposed parallel systolic-like array for the finite field multiplication over *GF(2m)* based on irreducible AOP. (a) The linear array structure. (b) Function of PE-1. (c) Function of PE-2. (d) Function of the *i*th regular PEs (PE-3 to PE-7).

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 require 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 $T = TX$, where TX is the delay of an XOR gate. One of the primary advantages of the proposed 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 proposed architecture in Fig. 2 can be expressed as $L = l + \lceil \log_2 s \rceil + 1$, (17) where l is the latency for the l PEs, $\lceil \log_2 s \rceil$ is the latency for the PAT and the constant 1 is for the ORC. According to (16), 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. 1 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 celland the last two PEs will have two redundant delays to wait for other PEs to complete their tasks. In this case, the latency of the multiplication would be 5 + 3 + 1 = 9 cycles, while the cycle time and throughput are the same as that of Fig. 3. Moreover, by substituting (16) in to (17), we have $L = \lceil m/s \rceil + \lceil \log_2 s \rceil + 1$. (18) The minimum latency for a specific m can be achieved by choosing the s value which minimize (18). In real practice, thevalue of l and s can be determined according to the latency requirement of the application.

B. Time-Multiplexed Systolic Structure

To have a hardware-efficient implementation, the computation of the rows of LUs of the 2-D DG in Fig. 2 can be time-multiplexed. The most straightforward and hardwareefficient way is to design the hardware for one row of Fig. 2 and time-multiplex it. This is referred to as TM-1 structure ,where 1 represents the time-multiplexed hardware for one row.The adder tree in Fig. 2 could be implemented by a finite field accumulator (FFA). The TM-1 structure of multiplier for *GF(2m)* based on irreducible AOP is shown in Fig. 4 for $m = 28$. It consists of seven PEs. The function of different PEs are shown in Figs. 4(b)–(e). PE-1 and PE-2 consists of only one AND cell each, while each of the regular PEs (PE-3 to PE-7) requires one AND cell and one XOR cell. Except PE-7 (the last PE), all other regular PEs consists of one reduction cell. At the front end of the linear array one multiple-reduction cell (MRC) is used to implement the function of nodes *S*. It reduces the input operand by order 7 in each cycle iteratively to generate successive input operands for the systolic array. The output of PE-7 is fed to the accumulator FFA to implement the function of the pipeline-adder-tree in a sequential manner. The register of FFA is initialized to zero and four successive outputs of PE-7 are added with the content of the register in 4 cycles to generate the desired $(m+1)$-bit word Y according to (13b). The function of ORC is exactly the same as that of output node O of Fig. 2(e). It performs the reduction of degree of Y from m to $(m-1)$ to produce the desired product word C according to (15). The input operand A is appended with a zero, and the $(m + 1)$ bit word P_{-1} thus generated is fed to PE-1 through a multiplexer, while the first 7 bits of operand B are fed to the seven PEs of the structure in a staggered manner. After 7 cycles PE-7 produces the output resulting from the

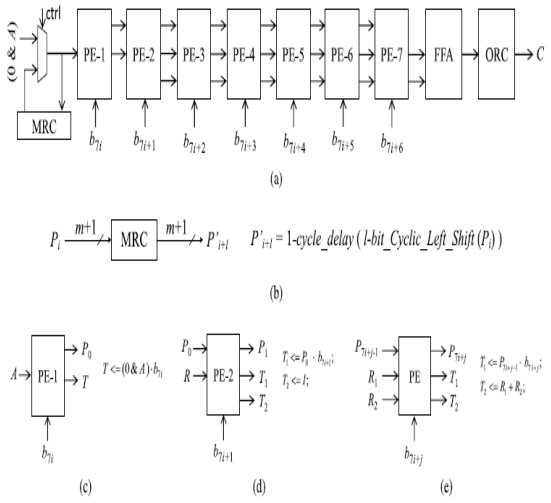


Fig. 4. The proposed time-multiplexed systolic-like array (TM-1) for the finite field multiplication over *GF(2m)* based on irreducible AOP. (a) The linear array structure. (b) Function of MRC. (c) Function f PE-1. (d) Function of PE-2. (e) Function of the *i*th regular PEs (PE-3 to PE-7).

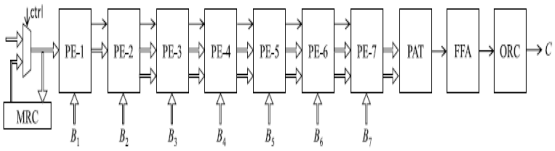


Fig. 5. The generalized time-multiplexed systolic-like array (TM-*n*) for the finite field multiplication over *GF(2m)* based on irreducible AOP. multiplication of operand *(0&A)* = *P*–1 with seven bits of *B*, which is added with the content of FFA (initialized to zero . The product of the reduced form of *P*7*i*–1 is fed to PE-1 during the *i* th cycle for the multiplication of that with the *i* th digit of seven bits of *B* by the array, followed by accumulation of the partial result in the FFA in the next three successive cycles for *i* = 1, 2, and 3. After four cycles of accumulation the conten of FFA is transferred to the ORC to reduce the degree by one to generate the product word *C*. The latency of this structure is 12 cycles (7 cycles for the PEs for the first partial result, 3 cycles for the next three successive partial results, 1 cycle for the FFA and 1 cycle for the ORC). The TM-1 structure is hardware-efficient at the cost of a moderate increase in latency, but it has relatively less throughput than the parallel structure. The TM-1 structure of Fig. 4, involves nearly 4 times less hardware and provides 4 times less throughput with an increase in latency by 2 cycles compared with the structure in Fig. 3. Moreover, by designing the hardware for *n* rows and timemultiplexing it, the TM-1 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. 2 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 precessed in time-multiplexed to be processed one after the next. Fig. 5 shows the generalized TM-*n* structure for AOP based finite field multiplication. The general structure of TM is similar to that of TM-1. In TM-*n*, all the PEs as well as the MRC need to precess the computation of *n* rows. Moreover, a PAT with _log2 *n*_ stages is needed to sum the results of *n* partial result Compared with TM-1, the TM-*n* structure provides *n* times more throughput as well as decreased latency, while it consumes *n* times more hardware. For example, if we use a TM structure to implement the DG in Fig. 2, the overall latency will be reduced to 11 cycles, where 7 cycles for the PEs for the first 2 partial results, 1 cycle for the next 2 partial results and 3 cycles for the PAT, FFA and ORC. The reduction of latency could be more significant for higher order fields with large *s* values.

MODIFIED BOOTH ALGORITHM:

Booth multiplication algorithm consists of three major steps as shown in structure of Booth algorithm figure that includes generation of partial product called as recoding, reducing the partial product in two rows, and addition that gives final product. modified Booth algorithm & for multiplication, we must know about each block of Booth algorithm for multiplication process. It is possible to reduce the number partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that , instead of shifting and adding for every column of multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by + (or) - 1,+ (or) -2,or 0,to obtain the same results.Radix-4 Booth encoder performs the process of encoding the multiplicand based on multiplier bits. It will compare 3 bits at a time with overlapping technique. Grouping starts from the LSB, and the first block only uses two bits of the multiplier and assumes a zero for the third bit. Radix-4 Booth algorithm is given below: 1. Extend the sign bit 1 position if necessary to ensure that n is even. 2. Append a 0 to the right of the LSB of the multiplier. 3. According to the value of each vector, each partial product will0,+y,y,+2y,-2y.

Bit position			Operation
i+1	I	i-1	
0	0	0	0*M
0	0	1	1*M
0	1	0	1*M
0	1	1	2*M
1	0	0	-2*M
1	0	1	-1*M
1	1	0	-1*M
1	1	1	0*M

RADIX16 MODIFIED BOOTH ENCODING ALGORITHM: Booth recoding was originally introduced when multiplication was implemented using a series of shift-add operations. By recoding, the number of 1's in the multiplier could be

reduced, and thereby the number of additions. The core idea is as follows:

Multiplier: $B=-b_{n-1}2^{n-1}+\sum_{i=0}^{n-2} b_i 2^i$

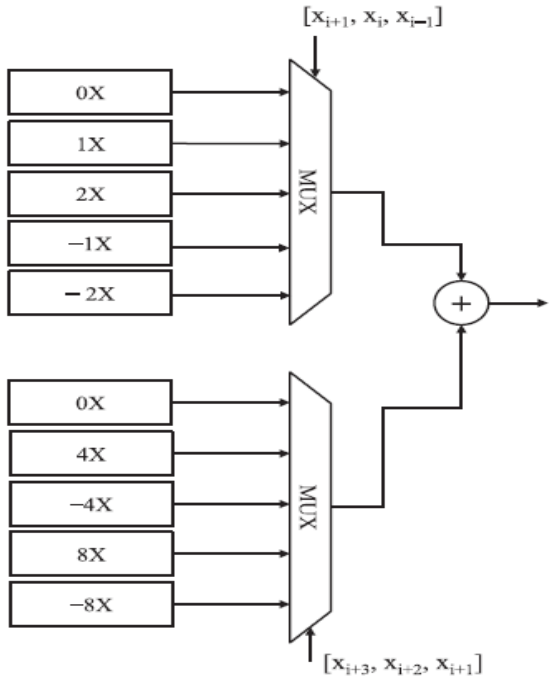
- Introduce new variables: $b'_i=-b_i+b_{i-1}$ for $i=0...n-1$ (assume $b_{-1}=0$).
- Compute $P=\sum_{i=0}^{n-1}(b'_i \times 2^i \times A)$
- Although this looks very similar to the normal product, note that any time $b_i=b_{i-1}$ there is no addition to be performed as the partial product will be zero. In the case of serial addition, these steps can be skipped, thus saving computation. To reduce the number of partia l products added while multiplying the multiplicand higher radix Booth Encoding algorithm is one of the most well known techniques used .Radix 16 Booth algorithm which scan strings of five bits with the algorithm given below (1) Extend the sign bit 1 p osition if necessary to ensure that n is even.

(2) Append a 0 to the right of the LSB of the multiplier.

(3) According to the value of each vector,each Partial Product will be 0,y,+2y,+3y,+4y,+5y,+6y,+7y,+8y, -8Y, - 7y, - 6y, - 5y, - 4y, -3y, - 2Y, - Y. The multiplicati on of y is done by shifting y by one bit to the left. Thus, in any case, in designing n bit parallel multipliers, only n/4 partial products are generated

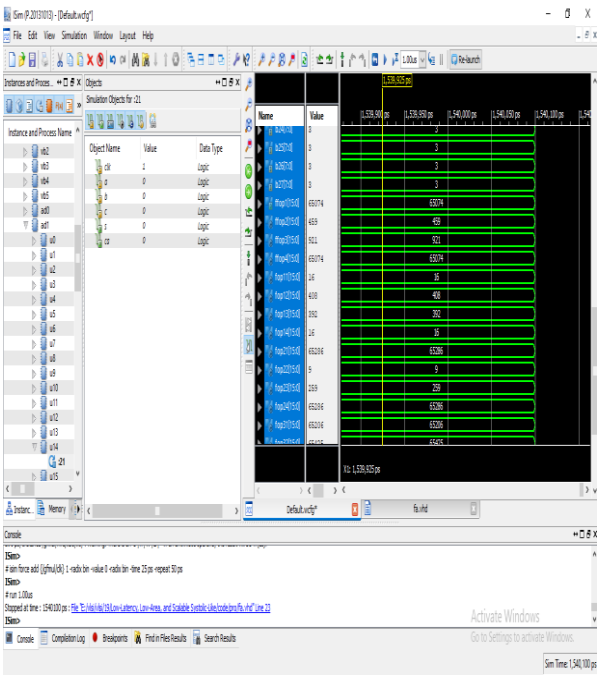
X_{i+3}	X_{i+2}	X_{i+1}	X_i	X_{i-1}	PP
0	0	0	0	0	0Y
0	0	0	0	1	1Y
0	0	0	1	0	1Y
0	0	0	1	1	2Y
0	0	1	0	0	2Y
0	0	1	0	1	3Y
0	0	1	1	0	3Y
0	0	1	1	1	4Y
0	1	0	0	0	4Y
0	1	0	0	1	5Y
0	1	0	1	0	5Y
0	1	0	1	1	6Y
0	1	1	0	0	6Y
0	1	1	0	1	7Y
0	1	1	1	0	7Y
0	1	1	1	1	8Y
1	0	0	0	0	-8Y
1	0	0	0	1	-7Y
1	0	0	1	0	-7Y
1	0	0	1	1	-6Y
1	0	1	0	0	-6Y
1	0	1	0	1	-5Y
1	0	1	1	0	-5Y
1	0	1	1	1	-4Y
1	1	0	0	0	-4Y
1	1	0	0	1	-3Y
1	1	0	1	0	-3Y
1	1	0	1	1	-2Y
1	1	1	0	0	-2Y
1	1	1	0	1	-1Y
1	1	1	1	0	-1Y
1	1	1	1	1	0Y

Radix16 modified booth encoding table



Radix16 modified booth encoder

RESULT:



CONCLUSION:

A powerful recursive detailing is recommended for the systeolic usage of sanctioned based limited field augmentation over G F(2 m) in view of 8 bit final AOP where cyclic-left-shift operations achieve the required modular degree reduction. Appropriate DGs are obtained from recursive formulation and from this are engineered systolic multipliers. In this we formulated the importance of Pipe lining for reliable speed operations in the irreducible AOP in the systolic processing element structures. Modified proposed multiplier results shows significant optimized parameters in both time and area scenarios.

REFERENCES:

1. J. Xie, P. Meher, and J. He, "Low-complexity multiplier for *G F(2m)* based on all-one polynomials," *IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst.*, vol. 21, no. 1, pp. 168–173, Jan. 2013.

2. J. Menezes, I. F. Blake, S. Gao, S. A. V. R. C. Mullin, and Yaghoobian, Eds., *Applications of Finite Fields*. Boston, MA, USA: Kluwer, 1993.

3. R. Lidl and H. Niederreiter, Eds., *Introduction to Finite Fields TheirApplication*. New York, NY, USA: Cambridge Univ. Press, 1986.

4. [Online]. Available: http://www.csrc.nist.gov/publications

5. S. K. Jain, L. Song, and K. K. Parhi, "Efficient semisystolic architectures for finite-field arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI)Syst.*, vol. 6, no. 1, pp. 101–113, Mar. 1998.

6. L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 19, no. 2, pp. 149–166, 1998.

7. P. K. Meher, "Systolic formulation for low-complexity serial-parallel implementation of unified finite field multiplication over *G F(2m)*," in *Proc. 18th IEEE Int. Conf. Appl.-Specific Syst., Archit. Process. (ASAP)*,Jul. 2007, pp. 134–139.

8. 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.

9. A. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over *G F(2m)*," *IEEETrans. Comput.*, vol. 53, no. 8, pp. 945–959, Aug. 2004.

10. W. Tang, H. Wu, and M. Ahmadi, "VLSI implementation of bit-parallel word-serial multiplier in *G F(2233)*," in *Proc. 3rd Int. IEEE-NEWCASConf.*, Jun. 2005, pp. 399–402.

11. H. Wu, "Low complexity bit-parallel multiplier for a class of finite fields," in *Proc. Int. Conf. Commun., Circuits Syst.*, vol. 4. Jun. 2006, pp. 2565–2568.

12. P. K. Meher, "High-throughput hardware-efficient digit-serial architec-ture for field multiplication over *G F(2m)*," in *Proc. 6th Int. Conf. Inf.Commun. Signal Process. (ICICS)*, Dec. 2007, pp. 1–5.

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

14. R. Katti and J. Brennan, "Low complexity multiplication in a finite field 1

15 A. Reyhani-Masoleh and M. A. Hasan, "A new construction of Massey–Omura parallel multiplier over GF (2m)," IEEE Trans Comput., vol. 51, no. 5, pp. 511–
520, May 2002.