Impact Factor (SJIF): 3.632

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 3, Issue 4, April-2016

PERFORMANCE ANALYSIS OF MULTIPLICATION AND INVERSION ALGORITHMS OVER GF(2^M) FOR CODING AND CRYPTOGRAPHIC APPLICATION

Ms. Jyoti N. Dindalkoppa[1], Mrs. R. H. Korti[2],

[1] *P.G. Student, Department of ECE, SDM College of Engineering and Technology , Dharwad*
[2]*Professor, Department of ECE, SDM College of Engineering and Technology , Dharwad*

Abstract – Finite field arithmetic logic is central in the implementation Of Reed-Solomon codes and in some cryptographic algorithms. There is a need for good multiplication and inversion algorithms that can be easily realized on VLSI chips. This paper presents a novel sequential Type-I optimal normal basis multiplier in GF(2^m) with a regular structure. The proposed multiplier is highly regular, modular, expandable and well-suited to VLSI implementation. A new normal basis inverter based on the proposed multiplier is also presented. The proposed inverter provides better time-area complexity than existing inverters as with large m.

Keywords--- Cryptography, Finite Field, Multiplication, Normal Basis, Multiplicative Inverse, VLSI

I. INTRODUCTION

Arithmetic over finite fields GF(2^m) has recently found many significant applications, including error correcting codes [1], cryptography [2], digital signal processing [3,4], switching theory [5] and pseudorandom number generation [6]. Addition, multiplication, exponentiation and inversion are the most important computations in finite field arithmetic. Addition can be easily implemented as XOR of the corresponding vectors. Multiplication typically requires more computational time than addition, and has more circuit complexity. Other important arithmetic operations, such as exponentiation, division, and multiplicative inversion, can be conducted by repeatedly applying the multiplication squaring algorithm. The finite field GF(2^m) is a number system containing 2^m elements. Its attractiveness in practical applications stems from the fact that each element can be represented by m binary digits. The practical application of error-correcting codes makes considerable use of computation in GF(2^m).Recent advances in secret communication, such as encryption and decryption of digital messages, also require the use of computation in GF(2^m) [4]. Hence, there is a need for good algorithms for doing multiplication and inversion in finite field. Different basis representations of field elements can be specified to simplify the implementation of arithmetic operations. Three major bases are standard, normal and dual basis. The standard basis multipliers [7] are extensively adopted, and result in efficient implementations of multipliers. As compared to the other two bases multipliers, the standard basis multipliers have a low design complexity, and their sizes are easier to extend to meet various applications owing to their simplicity, regularity and architectural modularity. The dual basis multipliers [8] require smaller chip areas than other two types. The major benefit of the normal basis multipliers [9] is that the squaring of an element is derived by cyclically shifting the binary representation. Thus, the normal basis multipliers are very effective for performing inverse, squaring and exponentiation operations However, the normal basis multipliers require basis conversion, since the field elements of GF(2^m) are represented using the standard basis. Massey and Omura formed the first normal basis multiplication algorithm [9]. The major flaws in the multiplier proposed by Massey Omura are its irregularity and lack of modularity, which mean that it cannot easily be extended .To eradicate this problem, this paper presents a novel sequential semi-systolic Type-I optimal normal basis multiplier with a space complexity of O(m) and features that are valuable for high-speed VLSI system design, such as regularity and modularity. The sequential multiplier iteratively determines the product of two elements with *m* bits in parallel. The products are accumulated after m clock cycles. A new normal basis inverter based on the new normal basis multiplier is also developed.

II. NORMAL BASIS REPRESENTATION

It is well known that there always exists a normal basis in the finite field GF(2^m) for all positive integers m. For an α € GF(2^m),{ α, α^2 , α^4 , ..., $\alpha^{2^{(m-1)}}$ } is called a normal basis of GF(2^m) over GF(2) if α, α^2 , α^4 , ..., and $\alpha^{2^{(m-1)}}$ are linearly independent. A normal basis always exists in the finite field GF(2^m) for all positive integers *m.* Each element A €GF(2^m) can be uniquely expressed as

$$A = a_0\alpha^{2^0} + a_1\alpha^{2^1} + a_2\alpha^{2^2} + \ldots + a_{m-1}\alpha^{2^{m-1}} \quad \ldots\ldots(1)$$

where $a_i \in \{0,1\}$ for $i = 0, 1, 2, \ldots, m-1$. If for all $0 \le i1, i2 \le m - 1$ and $i1 \ne i2$, there exist $j1, j2$ such that $A^{2^{i1}+2^{i2}} = A^{2^{j1}+2^{j2}}$ the basis is called optimal. Two commonly employed Optimal Normal Bases (ONBs) are defined as follows:

(1) Type-I ONB: $m+1$ is a prime p, and 2 is primitive modulo p.
(2) Type-II ONB: $2m+1$ is a prime p and either
  (a) 2 is primitive modulo p, or
  (b) $p \equiv 3 \pmod 4$ and the multiplicative order of 2 modulo p is m.

In this paper Type-I ONB is used t perform the multiplication and inversion.

III. THE PROPOSED NORMAL BASIS MULTIPLIER

Let $A$and B denote any two elements in GF(2^m), written as

$$A = a_0\alpha^{2^0} + a_1\alpha^{2^1} + a_2\alpha^{2^2} + \ldots + a_{m-1}\alpha^{2^{m-1}}, \text{ and}$$
$$B = b_0\alpha^{2^0} + b_1\alpha^{2^1} + b_2\alpha^{2^2} + \ldots + b_{m-1}\alpha^{2^{m-1}} \quad \ldots\ldots(2)$$

The product C of A and B is as follows:

C=A*B .....(3)

The major advantage of the normal basis representation is that an element in GF(2^m) is squared with a simple cyclic shift. However, multiplication in this basis appears to be more complex than in the other bases. Hence, normal basis multiplication requires basis conversion, to perform the multiplication in another basis. The normal basis N is expressed as.

$$N = \{\alpha, \alpha^2, \alpha^{2^2}, \ldots, \alpha^{2^{m-1}}\} \quad \ldots\ldots(4)$$

Let the generating polynomial $G(X)$ be an irreducible All-One-Polynomial [12] of degree m, where $m+1$ is relative prime to 2, and $G(X)$ is represented as

$$G(X) = 1 + X^1 + X^2 + \ldots + X^m \quad \ldots\ldots(5)$$

If α denotes the root of the G(X), then it has the following property

$\alpha^{m+1}=1$

then the normal basis N can easily be converted to the following shifted standard basis N':

$$N' = \{\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^m\} \quad \ldots\ldots(6)$$

Permutation P performs the following transformations for both A and B:

$$\begin{aligned} A &= a_0\alpha^{2^0} + a_1\alpha^{2^1} + a_2\alpha^{2^2} + \ldots + a_{m-1}\alpha^{2^{m-1}} \\ &= a_1'\alpha^1 + a_2'\alpha^2 + a_3'\alpha^3 + \ldots + a_m'\alpha^m, \\ B &= b_0\alpha^{2^0} + b_1\alpha^{2^1} + b_2\alpha^{2^2} + \ldots + b_{m-1}\alpha^{2^{m-1}} \\ &= b_1'\alpha^1 + b_2'\alpha^2 + b_3'\alpha^3 + \ldots + b_m'\alpha^m, \end{aligned} \quad \ldots\ldots(7)$$

Where

For i=0,1,2.....,m-1 and j=1,2,3,.....,m

$a'_j = a_i$

$b'_j = b_i$

and $j=2^i \bmod(m+1)$

Assuming that two elements A and B are represented by the shifted standard basis, the product C of A and B is calculated as

$$\begin{aligned} C &= A * B \\ &= (a_1'\alpha + a_2'\alpha^2 + a_3'\alpha^3 + \ldots + a_m'\alpha^m) * B \\ &= a_1'\alpha B + a_2'\alpha^2 B + a_3'\alpha^3 B + \ldots + a_m'\alpha^m B \end{aligned} \quad \ldots\ldots(8)$$

Each term in the equation (8) can be calculated as

(a)

$$
\begin{aligned}
&a_1'\alpha B \\
&= a_1'\alpha(b_1'\alpha + b_2'\alpha^2 + b_3'\alpha^3 + \ldots + b_m'\alpha^m) \\
&= a_1'(b_1'\alpha^2 + b_2'\alpha^3 + b_3'\alpha^4 + \ldots + b_m'\alpha^{m+1}) \\
&= a_1'(b_m' + b_1'\alpha^2 + b_2'\alpha^3 + b_3'\alpha^4 + \ldots + b_{m-1}'\alpha^m),
\end{aligned}
$$

$$
\begin{aligned}
&a_2'\alpha^2 B \\
&= a_2'\alpha(\alpha B) \\
&= a_2'\alpha(b_m' + b_1'\alpha^2 + b_2'\alpha^3 + b_3'\alpha^4 + \ldots + b_{m-1}'\alpha^m) \\
&= a_2'(b_m'\alpha + b_1'\alpha^3 + b_2'\alpha^4 + b_3'\alpha^5 + \ldots + b_{m-1}'\alpha^{m+1}) \\
&= a_2'(b_{m-1}' + b_m'\alpha + b_1'\alpha^3 + b_2'\alpha^4 + b_3'\alpha^5 + \ldots + b_{m-2}'\alpha^m),
\end{aligned}
$$

$$
\begin{aligned}
&a_3'\alpha^3 B \\
&= a_3'\alpha(\alpha^2 B) \\
&= a_3'\alpha(b_{m-1}' + b_m'\alpha + b_1'\alpha^3 + b_2'\alpha^4 + b_3'\alpha^5 + \ldots + b_{m-2}'\alpha^m) \\
&= a_3'(b_{m-1}'\alpha + b_m'\alpha^2 + b_1'\alpha^4 + b_2'\alpha^5 + b_3'\alpha^6 + \ldots + b_{m-2}'\alpha^{m+1}) \\
&= a_3'(b_{m-2}' + b_{m-1}'\alpha + b_m'\alpha^2 + b_1'\alpha^4 + b_2'\alpha^5 + b_3'\alpha^6 + \ldots + b_{m-3}'\alpha^m)
\end{aligned}
$$

Therefore, the term $a_i \alpha^i B$ for $i = 1, 2, 3, \ldots, m$ is computed as

$$
\begin{aligned}
&a_i'\alpha^i B \\
&= a_i'\alpha(\alpha^{i-1} B) \\
&= a_i'\alpha(b_{m-i+2}' + b_{m-i+3}'\alpha + b_{m-i+4}'\alpha^2 + \ldots + b_m'\alpha^{i-2} + b_1'\alpha^i + b_2'\alpha^{i+1} \\
&\qquad + \ldots + b_{m-i+1}'\alpha^m) \\
&= a_i'(b_{m-i+1}' + b_{m-i+2}'\alpha + b_{m-i+3}'\alpha^2 + b_{m-i+4}'\alpha^3 + \ldots + b_m'\alpha^{i-1} \\
&\qquad + b_1'\alpha^{i+1} + b_2'\alpha^{i+2} + \ldots + b_{m-i}'\alpha^m)
\end{aligned}
\qquad \ldots\ldots(9)
$$

By summing up the corresponding terms in the above equation for $1 \le i \le m$, and one extra term, each term of the product C is calculated using

$$
\begin{aligned}
&c_0' = (a_0'b_0' + a_1'b_m' + a_2'b_{m-1}' + a_3'b_{m-2}' + \ldots + a_m'b_1') \mod 2, \\
&c_1' = (a_0'b_1' + a_1'b_0' + a_2'b_m' + a_3'b_{m-1}' + \ldots + a_m'b_2') \mod 2, \\
&c_2' = (a_0'b_2' + a_1'b_1' + a_2'b_0' + a_3'b_m' + \ldots + a_m'b_3') \mod 2, \\
&c_3' = (a_0'b_3' + a_1'b_2' + a_2'b_1' + a_3'b_0' + a_4'b_m' + a_5'b_{m-1}' + \ldots + a_m'b_4') \mod 2, \\
&\ldots \\
&c_{m-1}' = (a_0'b_{m-1}' + a_1'b_{m-2}' + a_2'b_{m-3}' + \ldots + a_{m-1}'b_0' + a_m'b_{m-1}') \mod 2, \\
&c_m' = (a_0'b_m' + a_1'b_{m-1}' + a_2'b_{m-2}' + \ldots + a_{m-1}'b_1' + a_m'b_0') \mod 2
\end{aligned}
\qquad \ldots\ldots(10)
$$

The final result C = A* B is given by

$$C = c_0\alpha^{2^0} + c_1\alpha^{2^1} + c_2\alpha^{2^2} + \ldots + c_{m-1}\alpha^{2^{m-1}} \qquad \ldots\ldots(11)$$

Figure 1 illustrates the hardware implementation of the proposed algorithm. Permutations *P*1 and *P*2 belong to permutation *P*, and permutation *P*3 belongs to the inverse permutation *P*-1. The functions of *P*1, *P*2 and *P*3, each with m inputs and m outputs are defined by

Permutations p1 and p2 with inputs I_j and outputs O_i

$O_i = I_j$

$I = 2^j \mod(m+1)$ for i = 1,2,3.....,m and j = 0,1,2,..... ,m-1

Inputs for p1 are given as

$I_i = b_i$ where, $0 \le i \le m-1$

Outputs for *P*1 are given by
$b_i' = O_i$ where, $1 \le i \le m$
apply $b_0' = 0$ and b_0 directly to flip flop D_0

Inputs for *P*2 are given as
$I_i = a_i$ where, $0 \le i \le m-1$

Outputs from *P*2 are given as
$a_i' = O_i$ where, $1 \le i \le m-1$
apply $a_0 = 0$ and a_0 directly in to s_0

Inverse permutation p3 with inputs I_i and outputs O_j
$O_j = I_i$
$j = 2^i \bmod(m+1)$

The final result *C* is obtained through permutation *P*3. The proposed normal basis multiplier needs m+1 2-input AND gates, 2m+1 2-input XOR gates and 3m+3 1-bit flip-flops.The proposed sequential normal basis multiplier is regular and expandable, and is therefore naturally suited to VLSI implementation.

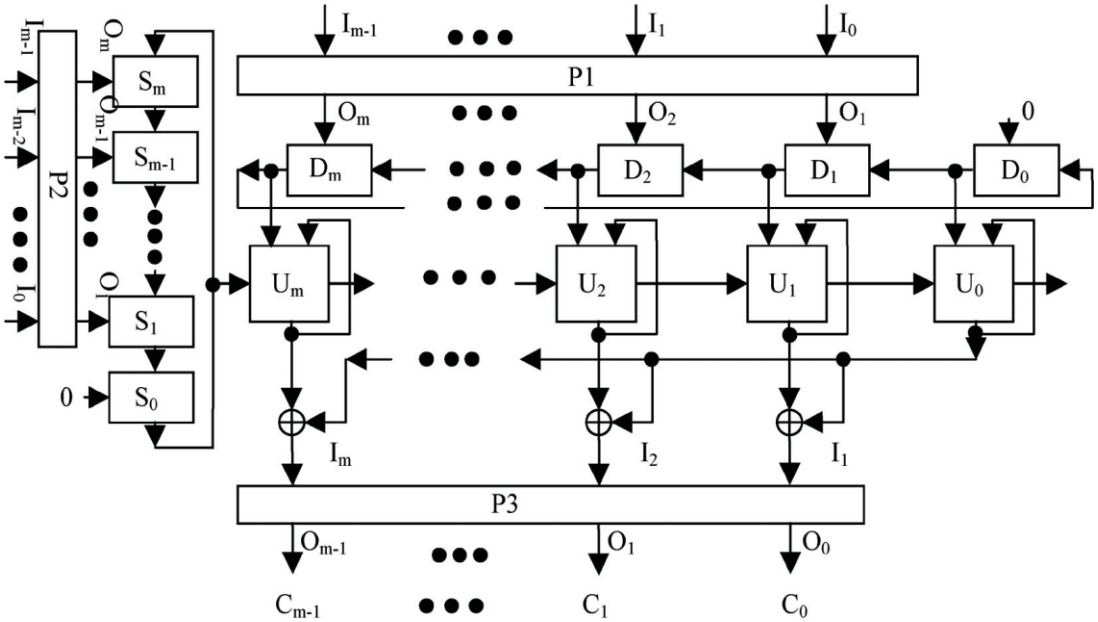

Figure 1. The proposed normal basis multiplier in GF(2^m).

IV. THE PROPOSED NORMAL BASIS MULTIPLICATIVE INVERTER

Multiplicative inversion is highly complex and most studied finite field arithmetic operation. A novel multiplicative inversion is developed based on the proposed normal basis multiplier.
From Fermat's theorem, for every $B \in GF(2^m)$, $B^{2^m} = B$ yielding

$$B^{-1}$$
$$= B^{2^m-2}$$
$$= B^{2+2^2+2^3+\ldots+2^{m-1}}$$
$$= B^2 B^{2^2} B^{2^3} \ldots B^{2^{m-1}}$$
$$= B^2 (B^2)^2 ((B^2)^2)^2 \ldots \overbrace{((\ldots((B)^2)^2)^2\ldots)^2}^{m-1} \quad \ldots\ldots\ldots\ldots(12)$$

Figure 3 shows the hardware implementation based on Eq. (12). The shift register *T*, which comprises m flip-flops, responds to the squaring computation of B^2, $(B^2)^2$, …, and $\overbrace{(\ldots(B^2)^2\ldots)^2}^{m-1}$. Permutations *P*1 and *P*2 belong to permutation *P* and P^{-1}, respectively. The proposed algorithm for multiplicative inverse is decribed below.

Algorithm:
/*computing B^{-1} */

Step 1: Initialization
(1) Reset all 1-bit latches in cells U_i for $0 \le i \le m$ to 0s.
(2) Load operand *B* into shift register *T*.

Step 2: Deriving B^2
(1) Shift *T* to left by one bit.
(2) $D_0 = 0$, load D with *T* through permutation *P*1.
(3) Do not shift D.
(4) $S_0 = 1$
(5) Load final B^2 into shift register *S*.
(6) $S_0 = 0$

Step 3: Squaring and multiplication
(1) Shift *T* to left by one bit.
(2) $D_0 = 0$; load *D* with *T* through permutation *P*1.
(3) Shift *D* and *S* one bit for each clock cycle. After m+1 clock cycles, obtain *D*S* and store it in *S*.

Step 4: Repeat Step 3 m-3 times. Determine the final result of *B*-1 from the output of permutation *P*2.

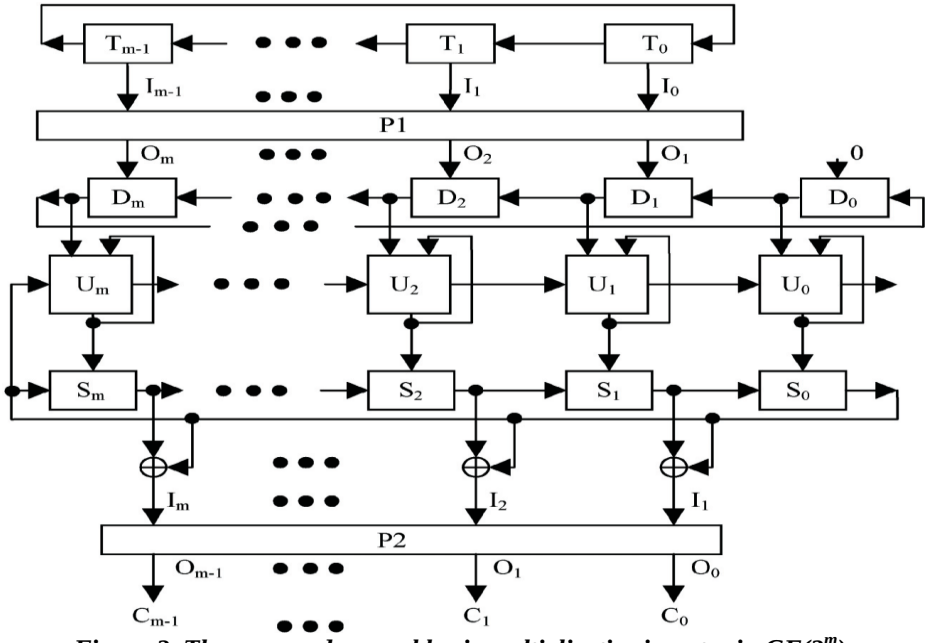

Figure 2. The proposed normal basis multiplicative inverter in GF(2^m).

The proposed inverter is regular and modular, making it very attractive for VLSI implementation. The proposed inverter provides better time-area complexity for the larger value of m.

IV. RESULTS

In this paper, a normal basis multiplier and a multiplicative inverter is implemented using VERILOG coding technique. For simulation, synthesis and timing analysis Xilinx 14.5 is used. The experiment results are divided into following parts which are simulation results as waveform, RTL schematic, utilization of resources , area in terms of LUTs & delay. Steps for simulation in any coding is development of the algorithm to be coded..Once the algorithm is ready, coding is started and different modules are developed. These modules are independently checked for errors and later they are assembled to form the exact code. The whole code is also checked for errors and logic is checked by simulating the code.

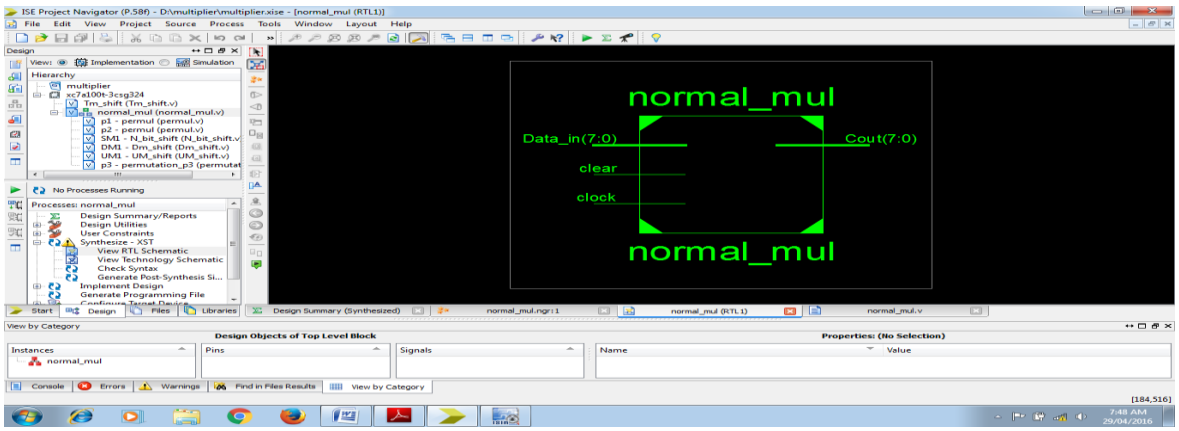

Figure 3. Symbol of normal basis multiplier.

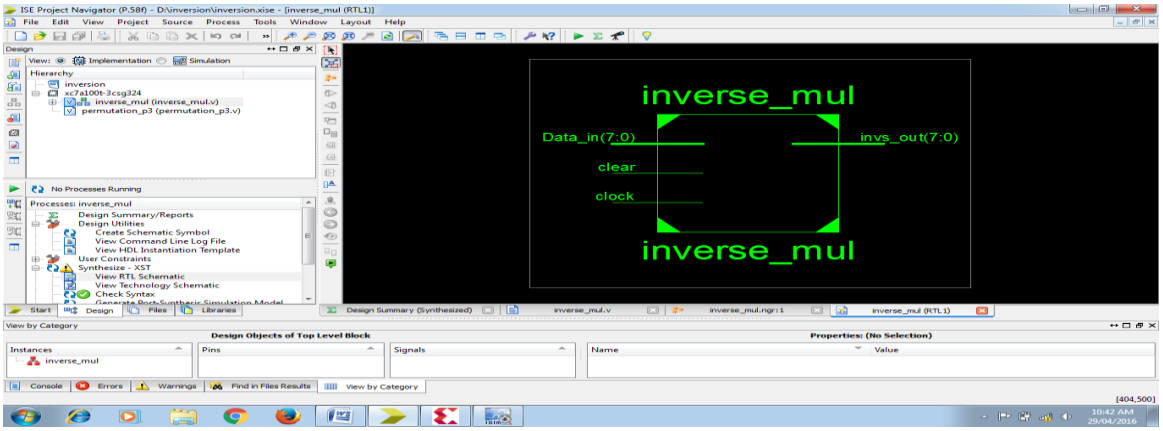

Figure 4. Symbol of normal basis multiplicative inverse.

A. Simulation Results

Simulation is done on Xilinx ISE Simulator. Here 8 bit inputs are used by giving any values for 8 bit input the outputs for both the multiplication and inversion can be obtained after certain clock cycles.

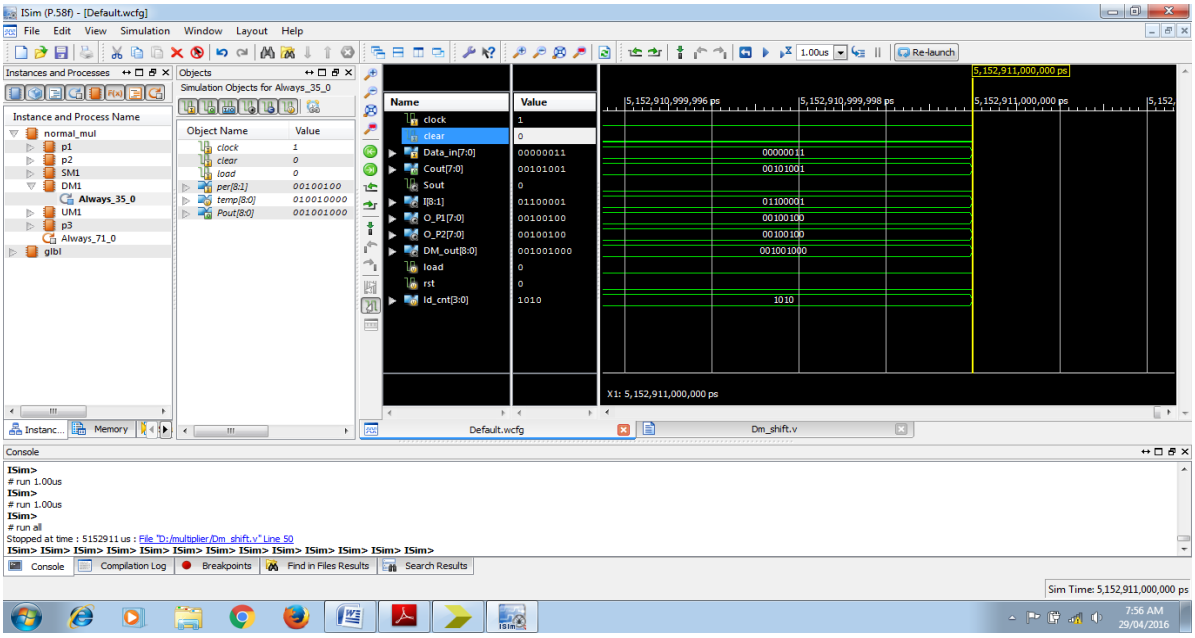

Figure 5. Waveform of normal basis multiplier.

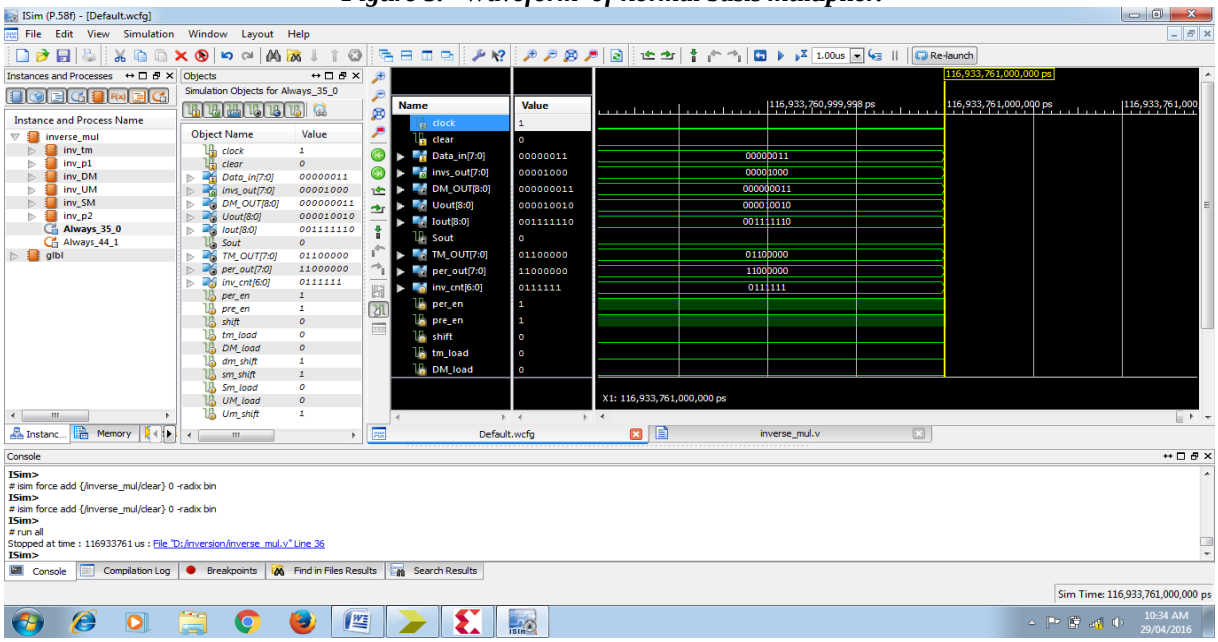

Figure 6. Waveform of normal basis multiplicative inverse.

B. Utilization of Resources

In any design the utilization of resources is most important, it defines the utility of device. For the above LDPC design utilization is as follows:

Table 1. Utilization of resources for normal basis multiplier

Logic Utilization	Used	Available
Number of Slice Registers	79	126800
Number of Slice LUT's	77	63400
Number of fully used LUT- FF pairs	63	93

Table 2. Utilization of resources for normal basis multiplicative inverse

Logic Utilization	Used	Available
Number of Slice Registers	110	126800
Number of Slice LUT's	142	63400
Number of fully used LUT- FF pairs	96	156

V. CONCLUSION

In this paper a novel sequential semi systolic Type-I ONB normal basis multiplier is presented and it is implemented by using verilog code in Xilinx. The proposed multiplier is regular, expandable and its easily realizable by using existing VLSI technology. A new normal basis multiplicative inverter is developed which is based on the proposed multiplier.The proposed inverter provides better time-area complexity than existing inverters for large values of m.

ACKNOWLEDGEMENT

I am highly obliged to Department of Electronics and Communication Engineering, SDM College of Engineering and Technology. I express my warm thanks to my guide. Prof. R.H.Korti for their valuable instructions ,guidance , providing me with the facilities being required and conductive conditions for my project.

REFERENCES

[1] Che Wun Chiou, Chiou-Yng Lee and Yun-Chi Yeh," Sequential Type-I Optimal Normal Basis Multiplier and Multiplicative Inverse in GF(2^m)", Tamkang Journal of Science and Engineering, Vol. 13, No. 4, pp. 423_432 (2010)
[2] MacWilliams, F. J. and Sloane, N. J. A., "The Theory of Error-Correcting Codes", Amsterdam: North-Holland(1977).
[3] Lidl, R. and Niederreiter, H., "Introduction to Finite Fields and Their Applications", New York: Cambridge Univ. Press (1994).
[4] Blahut, R. E.,"Fast Algorithms for Digital Signal Processing",Reading, Mass.: Addison-Wesley (1985).
[5] Reed, I. S. and Truong, T. K., "The Use of Finite Fields to Compute Convolutions," IEEE Trans. Information Theory, Vol. IT-21, pp. 208_213 (1975).2008.
[6] Wang, C. C. and Pei, D., "A VLSI Design for Computing Exponentiation in GF(2^m) and Its Application to Generate Pseudorandom Number Sequences," IEEE Trans. Computers, Vol. 39, pp. 258_262 (1990).
[7] Bartee, T. C. and Schneider, D. J., "Computation with Finite Fields," Information and Computing, Vol. 6, pp.79_98 (1963).
[8] Berlekamp, E. R., "Bit-Serial Reed-Solomon Encoders,"IEEE Trans. Information Theory, Vol. IT-28, pp. 869-874 (1982).
[9] Massey, J. L. and Omura, J. K., "Computational Method and Apparatus for Finite Field Arithmetic," U.S.Patent Number 4,587,627, May (1986).
[10] Reyhani-Masoleh, A., "Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases," IEEE Trans. Computers, Vol. 55, pp.34_47 (2006).
[11] Reyhani-Masoleh, A. and Hasan, M. A., "Low Complexity Sequential Normal Basis Multipliers over GF(2^m)," Proc. 16th IEEE Symposium on Computer Arithmetic, Vol. 16, pp. 188_195 (2003).
[12] Chiou, C. W. and Lee, C. Y., "Multiplexer-Based Double-Exponentiation for Normal Basis of GF(2^m),"Computers & Security, Vol. 24, pp. 83_86 (2005).
[13] Charles C. Wang,T. K. Truong,Howard M. Shao, Leslie J. Deutsch, Jim K. Omura, And Irving S.,"VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)," IEEE Transactions On Computers, Vol. C-34, No. 8, August 1985
[14] LakhendraKumar, Dr. K. L. Sudha," Implementation of Galois Field Arithmetic Unit on FPGA," IJIRCCE, Vol. 2, Issue 6, June 2014