EFFICIENT IMPLEMENTATION OF BIT PARALLEL FINITE FIELD MULTIPLIERS

Ajitha.S.S[1], Retheesh.D[2]

[1]*ME, Department of ECE, St.Xavier's Catholic College of Engineering, Tamilnadu, India*
[2]*Assistant Professor, Department of CSE, Saveetha Engineering College, Tamilnadu, India*

Abstract

Arithmetic in Finite/Galois field is a major aspect for many applications such as error correcting code and cryptography. Addition and multiplication are the two basic operations in the finite field GF (2^m).The finite field multiplication is the most resource and time consuming operation. In this paper the complexity (space) analysis and efficient FPGA implementation of bit parallel Karatsuba Multiplier over GF (2^m) is presented. This is especially interesting for high performance systems because of its carry free property. To reduce the complexity of Classical Multiplier, multiplier with less complexity over GF (2^m) based on Karatsuba Multiplier is used. The LUT complexity is evaluated on FPGA by using Xilinx ISE 8.1i.Furthermore,the experimental results on FPGAs for bit parallel Karatsuba Multiplier and Classical Multiplier were shown and the comparison table is provided. To the best of our knowledge, the bit parallel karatsuba multiplier consumes least resources among the known FPGA implementation.

Keywords: *Classical Multiplier, Cryptograph, FPGA, Galois field, Karatsuba Multiplier*

----------***----------

1. INTRODUCTION

A finite field or Galois field is a field that contains only finitely many elements. The finite fields are classified by size. This classification specifies the order of the field. Notations for the finite fields are GF (p^m) or F_p^m, where the letters "GF" stand for "Galois field". The order or cardinal or number of elements, of a finite field is of the form p^m, where p is a prime number called the characteristic of the field and m is a positive integer called the dimension of the field. Finite field arithmetic operations in GF (2^m) were frequently desired in coding theory, cryptography, digital signal processing. Coding theory is an approach to various science disciplines such as information theory, electrical engineering, data transmission, mathematics and science, which helps design efficient and reliable data transmission methods so that redundancy can be removed and errors corrected.

Applications of cryptography include ATM cards, computer passwords and e-commerce. Cryptography is the practice and study of hiding information. Modern cryptography intersects the disciplines of mathematics, computer science and engineering. In these applications, multiplication is the most common arithmetic. Thus it is desirable to design hardware-efficient multiplier for GF (2^m) to meet the real time requirement with minimum hardware complexity.

There are three popular types of bases over finite fields: polynomial basis (PB), normal basis (NB) and dual basis (DB). Basis is a set of vectors that, in a linear combination, can represent every vector in a given vector space. Polynomial basis is a mathematical function that is the sum of a number of terms. Normal basis in field theory is a special kind of basis for Galois extensions of finite degree, characterized as a forming a single orbit for the Galois group. Dual basis is a set of vectors that forms a basis for the dual space of a vector space. One advantage of the normal basis is that the squaring of an element is computed by a cyclic shift of the binary representation. The dual basis multipliers require less chip area than other two types.

The polynomial basis multipliers are widely used and lead to efficient implementations of multipliers. As compared to other two bases multipliers, the polynomial basis multipliers have low design complexity and their sizes are easier to extend to meet various applications due to their simplicity, regularity, and modularity in architecture. It appears that polynomial multipliers for classes of trinomials still achieve the lowest circuit complexity

Arithmetic operations such as addition and multiplication are the two basic operations in the finite field GF (2^m). Addition in GF (2^m) is easily realized using m two-input XOR gates while multiplication is costly in terms of gate count and time delay. The other operations of finite fields, such as exponentiation, division and inversion can be performed by repeated multiplications. As a result there is a need to have fast multiplication architecture with low complexity.

The hardware/software implementation efficiency of finite field arithmetic is measured in terms of the associated space and time complexities. The space complexity is defined as the number of XOR and AND gates needed for the

implementation of the circuit. The space and time complexities of a multiplier heavily depend on how the field elements are represented. Finite field multipliers with different bases of representation have been realized to be used for various applications. The polynomial basis multipliers are more efficient and more widely used compared with multipliers in the other bases of representations.

1.1 Related Work

C.Grabbe,M.Bednara,J.Teich, [1] presented four high performance GF(2233) multipliers for an FPGA realization and analyzed the time and area complexities. The finite field elements are represented as polynomial basis and normal basis. In polynomial basis, classical multiplier and Karatsuba multipliers were designed.The advantage of classical multiplier is regular structure and pipelined operation.The disadvantage is high space complexity. In Karatsuba multiplier the advantage is less number of gates are required.The normal basis multipliers are Massey-Omura and Sunar-Koc multiplier. The advantage of Massey-Omura is high flexibility and Sunar-Koc is total number of gates are reduced.

P.L.Montgomery,[2] presented Karatsuba Ofman algorithm for multiplying two polynomials. Here multiplication of 5-term, 6-term and 7-term polynomials are provided with scalar multiplication of 13,17 and 22.Using 6-term polynomial only leads to better asymptotic performance than standard karatsuba.

C.Paar, [3] presented a new bit parallel structure for a multiplier with low space complexity in Galois field is introduced. Finite field of $GF(2^n)$ is considered and field extension of $GF((2^n)^m)$. The field elements are represented in the canonical base or in standard basis. Field of the form $GF((2^n)^m)$ are referred as composite field. Karatsuba Ofman algorithm is used to multiply two polynomials effectively. Advantages are complexity is reduced by introducing the composite field. The main disadvantage is security is less and does not have a regular structure.

C.Rebeirno and D.Mukhopadhyay, [4] presented a hybrid technique which has a better area delay product. Masking strategies are introduced to prevent power based side channel attacks on the multiplier. SCAs are the biggest threat to modern cryptography systems. In basic recursive KM, the number LUTs required to combine the partial products is much lower but it cannot applied directly to ECC. The hybrid KM requires least resources as compared to other KMs used for elliptic curve arithmetic; also it has a unique architecture. Demerits are it is not efficient for FPGA platform as the number of utilized LUTs is 65%.

A.Reyhani-Masoleh and A.Hasan, [5] presented a new bit parallel structure for the polynomial basis multiplication which is applicable to all type of irreducible binary polynomial. The main advantage of this new formulation is that it can be used with any field defining irreducible polynomial. Then a bit parallel hardware architecture generalization is provided. The architecture consist of two parts IP network and Q network. The space and time complexities are analysed as a function of reduction matrix Q. the main advantage is only fewer number of lines are required on the bus.

F.Rodriguez and C.K.Koc,[6] presented the Karatsuba-Ofman Algorithm in which the degree of defining the irreducible polynomial can be arbitrarily selected by the designer allowing the usage of prime degrees. Here finite field and composite field are considered. Composite multiplication is performed by n-bit Karatsuba multiplier. The main advantage is number of multiplication is reduced. The composite field multiplication is performed by binary Karatsuba multipliers. The advantage is improved gate complexity .The disadvantage is wastage of several arithmetic operation.

B.Sunar,[7] presented the subquadratic complexity multipliers for even characteristic field extension. A short convolution algorithm named Winograd short convolution algorithm were designed to improve the space and time complexity. A certain Winograd short convolutional algorithm is essentially identical to the Karatsuba algorithm. The merits of Winograd techniques are it can be easily built for any desired length; it is simple and uniform construction. The main disadvantages are appears to have less structure and cause additional wire delay in VLSI implementation.

A.Weimerskirch and C.Paar, [8] presented the classical Karatsuba algorithm for polynomial multiplication. Three methods considered are digital method, Fast Fourier transform method and Karatsuba method. The Karatsuba algorithm is derived in two ways namely Chinese Remainder Theorem and Simple Algebraic Transform KA is applied recursively if the degree of polynomial is 2^i, where $i>1$ is a positive integer. Advantage- squaring the polynomial is easily performed; adding a dummy coefficients the complexity is reduced. Disadvantage is a number of intermediate results have been stored due to the recursive nature of KA.

J.VonzurGathen and J.Shokrollahi, [9] presented different possibilities for implementing the Karatsuba multiplier for polynomials over F_2 on FPGA. Classical multiplier, Karatsuba multiplier and a hybrid design were provided. The Karatsuba multiplier has the lowest crossover point with the classical algorithm. In hardware, the algorithmic and platform dependent optimizations yield efficient designs. The resources usage of polynomial multipliers is decreased by using both the algorithmic and platform dependent method. The hybrid design is used to minimize the total arithmetic cost and results in significant area savings.

G.Zhou, H.Michalik and L.Hinsenkamp, [10] addresses the efficient and high throughput implementations of AES-GSM optimized for FPGAs. The two main components in GCM are an AES engine and a finite field multiplier over GF (2^{128}).The complexity analysis presented is based on FPGA primitives (LUTs). Modular multiplication consists of two steps: first a classical multiplication and then a modular reduction. The straight forward multiplier is used to get speed efficient design while a Karatsuba multiplier is used to get an area efficient design. Merits are reduced hardware complexity and high throughput.

2. FINITE FIELD ARITHMETIC

Arithmetic in a finite field is different from standard integer arithmetic. There are limited numbers of elements in the finite field; all operations performed in the finite fields result in an element within that field. While each finite field itself is not infinite, there are infinitely many different finite fields; their number of elements is necessarily of the form p^n. and the two finite fields of the same size are isomorphic. An element α in *GF* (2^m) can be represented as a polynomial, where $\alpha_i \in GF(2^m)$.

$$A = \sum_{i=0}^{m-1} \alpha_i x^i$$

Addition of two elements in *GF* (2^m) is performed as polynomial addition in

GF (2^m)

$$\sum_{i=0}^{m-1} a_i z^i + \sum_{i=0}^{m-1} b_i z^i = \sum_{i=0}^{m-1} (b_i + a_i) z^i$$

Where + is XOR operation.

2.1 Addition and Multiplication In Finite Field

Addition and subtraction are performed by adding or subtracting two of these polynomials together, and reducing the result modulo the characteristic. In a finite field with characteristic 2, addition and subtraction are identical, and are accomplished using the XOR operator. When two polynomials are added, each term is added independently; there is no concept of a carry from one term to another.

Thus, Polynomial:

$(x^6 + x^4 + x + 1) + (x^7 + x^6 + x^3 + x) = x^7 + x^4 + x^3 + 1$

Binary: {01010011} + {11001010} = {10011001}

Hexadecimal: {53} + {CA} = {99}

Notice that under regular addition of polynomials, the sum would contain a term $2x^6$, but that this term becomes $0x^6$ and is dropped when the answer is reduced modulo 2.

In binary representation, the coefficients can be only 1 or 0.When adding the coefficients, the following rule applies:

- 0+0=0
- 0+1=1
- 1+0=1
- 1+1=0(there is no carry)

Table-1: Polynomial Addition

$\mathbf{p_1}$	$\mathbf{p_2}$	$\mathbf{p_1} + \mathbf{p_2}$ (normal algebra)	$\mathbf{p_1} + \mathbf{p_2}$ in GF(2^n)
$x^3 + x + 1$	$x^3 + x^2$	$2x^3 + x^2 + x + 1$	$x^2 + x + 1$
$x^4 + x^2$	$x^6 + x^2$	$x^6 + x^4 + 2x^2$	$x^6 + x^4$
$x + 1$	$x^2 + 1$	$x^2 + x + 2$	$x^2 + x$
$x^3 + x$	$x^2 + 1$	$x^3 + x^2 + x + 1$	$x^3 + x^2 + x + 1$
$x^2 + x$	$x^2 + x$	$2x^2 + 2x$	0

Table-1 shows both the normal algebraic sum and the characteristic 2 finite field sum of a few polynomials:

In computer science applications, the operations are simplified for finite fields of characteristic 2, also called GF(2^n) Galois fields, making these fields especially popular choices for applications. The same logic that made addition become XOR also applies to subtraction. The exclusive OR operation is easier for digital logic than binary additions.

Multiplication in a finite field is multiplication modulo an irreducible reducing polynomial used to define the finite field. (i.e., it is multiplication followed by division using the reducing polynomial as the divisor—the remainder is the product.) The symbol "•" may be used to denote multiplication in a finite field. Example: Rijndael's finite field

Rijndael uses a characteristic 2 finite field with 8 terms, which can also be called the Galois field GF (2^8). It employs the following reducing polynomial for multiplication:

$$x^8 + x^4 + x^3 + x + 1.$$

For example, {53} • {CA} = {01} in Rijndael's field because

$(x^6 + x^4 + x + 1)(x^7 + x^6 + x^3 + x) =$

$x^{13} + x^{12} + x^9 + \mathbf{x^7} + x^{11} + x^{10} + \mathbf{x^7} + x^5 + x^8 + \mathbf{x^7} + x^4 + x^2 + \mathbf{x^7} + x^6 + x^3 + x =$

$x^{13} + x^{12} + x^9 + x^{11} + x^{10} + x^5 + x^8 + x^4 + x^2 + x^6 + x^3 + x =$

$x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x$

And

$x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x$ modulo $x^8 + x^4 + x^3 + x + 1$ = (11111101111110 mod 100011011) = 1, which can be performed through long division method

2.2 Classical Multiplier

The Classical multiplier is the simplest multiplier to perform finite field multiplication. It is also called as paper and pencil method or traditional method. To perform the classical multiplication it requires only AND gates and XOR gates. The number of AND gates required is n^2 and $(n-1)^2$ XOR gates, where n is bit depth. The total gate complexity is $2n^2-1$ and the time complexity is $T_{AND}+(\log_2 n)T_{XOR}$. Figure 1 shows the calculation of the product of two 4-bit integer numbers given by A3A2A1A0 (multiplicand) and B3B2B1B0 (multiplier).

				A3	A2	A1	A0
				B3	B2	B1	B0
				A3.B0	A2.B0	A1.B0	A0.B0
			A3.B1	A2.B1	A1.B1	A0.B1	
		A3.B2	A2.B2	A1.B2	A0.B2		
	A3.B3	A2.B3	A1.B3	A0.B3			
S6	S5	S4	S3	S2	S1	S0	

Fig-1: 4-bit multiplication

Each of the ANDed terms is referred to as a partial product. The final product is obtained by summing each column of partial products and is implemented using half adders.. If carry comes, it must be propagated from the right to the left across the columns. Adder that accepts a carry from the right must be a full adder. 4-bit classical multiplier is shown in figure 2.

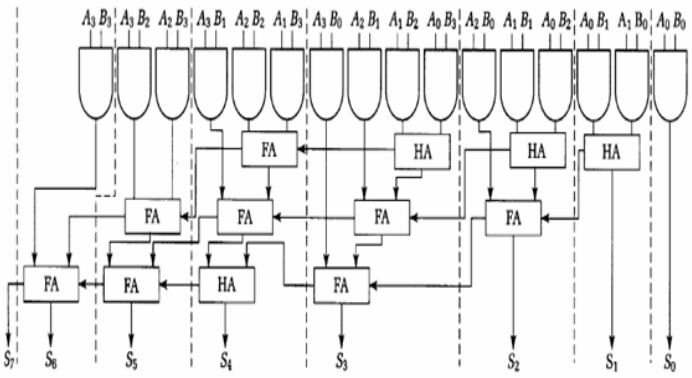

Fig -2 4-bit Classical Multiplier

The classical multiplier consists of AND gates, Full Adders and Half adders. The 16 AND gates forms the sixteen partial products. It is formed by ANDing all combinations of the four multiplier bits with the four multiplicand bits. The column sums are obtained using a combination of half and full adders.

The half adder blocks accept two bits to be added from the top, carry out exits from the left of each block. The output from the bottom of a block is the sum. The full adder accepts two bits to be added from the top, any carry in from the right and carryout exist from left of each block. The bottom of each block gives the output. The least significant output bit, S0 is computed as the product of two input bits A0 and B0.This operation cannot generate a carry out. The next output bit, S1, involves the sum of two partial products. A half adder is used to form the sum since there can be no carry in from the first column.

The third output bit, S2, is formed from the sum of three (1-bit) partial products plus a possible carry in from the previous bit. This operation requires two cascaded adders to sum the four possible input bits (three partial products and one possible carry in from the right). The remaining output bits are formed similarly.

2.3 Karatsuba Multiplier

The Karatsuba algorithm is an efficient multiplication algorithm. It reduces the multiplication of two n-digit numbers to at most single digit multiplications. It is the fast multiplication algorithm and at the same time it is the fast computational algorithm. It uses the technique divide and conquer technique. That is Karatsuba algorithm is applied to large degree polynomials by splitting it into a lower and an upper half. The algorithm becomes recursive if its applied again to multiply these polynomial half's.

The next iteration step splits these polynomial again in half. The algorithm eventually terminates after t-steps. In the final step the polynomial degenerates into single coefficients. This recursive splitting of polynomials and the special reassembling of the partial products ,drastically reduces the number of AND gates to $n^{\log_2 3}$ or $n^{1.59}$ but the n umber of XOR gates increased to $6n^{\log_2 3}-8n+2$ which is very efficient when the polynomials become large. But in GF(2^n), multiplication is a quite expensive operation and an addition can be performed at nearly no costs(since an XOR is very small on an FPGA and no carry bits exist) .

For the modular multiplier, the straight forward multiplication is used to get a speed efficient design, while the karatsuba algorithm is used to get an area efficient design. But by combining classical and Karatsuba, we get a high performance and highly efficient multipliers. Squaring can be easily performed by applying Karatsuba algorithm.

In general consider two polynomials of degree m-1

$$A = \sum_{i=0}^{m-1} \alpha_i x^i = \sum_{i=m/2}^{m-1} \alpha_i x^i + \sum_{i=0}^{m/2-1} \alpha_i x^i$$

$$= x^{m/2} \sum_{i=0}^{m/2-1} \alpha_i + m/2 x^i + \sum_{i=0}^{m/2-1} \alpha_i x^i = x^{m/2} A^H + A^L$$

and

$$B = \sum_{i=0}^{m-1} b_i x^i = \sum_{i=m/2}^{m-1} b_i x^i + \sum_{i=0}^{m/2-1} b_i x^i$$

$$= x^{m/2} \sum_{i=0}^{m/2-1} b_i + m/2 x^i + \sum_{i=0}^{m/2-1} b_i x^i = x^{m/2} B^H + B^L$$

The polynomial product is

$$C = x^m A^H B^H + (A^H B^L + A^L B^H) x^{(m/2)} + A^L B^L.$$

To perform an n-bit multiplication we need an algorithm that divides the n-bit multiplication into several one bit multiplications, which are the only multiplications that can be computed directly (i.e., by an AND-gate). With Karatsuba's divide and conquer multiplication algorithm, a multiplication of two n-bit polynomials can be computed with three n=2-bit multiplications and some additions (which are XOR's in our case) to determine interim results and accumulate the final result. If n is four or more, the three multiplications in Karatsuba's basic step involve operands with less than n digits. Therefore, those products can be computed by recursive calls of the Karatsuba algorithm. Figure 3 denotes 1-bit polynomial karatsuba multiplier. "."denotes multiplication in finite field.

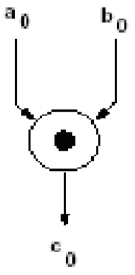

Fig-3: one bit polynomial Karatsuba multiplier

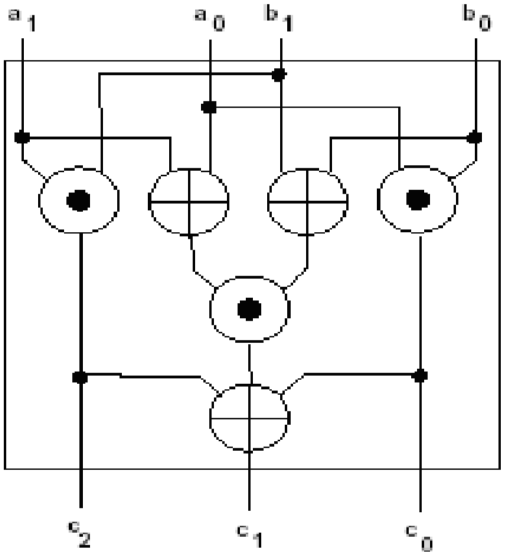

Fig-4: 2- bit polynomial Karatsuba multiplier

Figure 4 shows the two bit polynomial Karatsuba multiplier KM2.Here the dot represents the finite field multiplication (AND gate) and the plus represents the addition(XOR gate).The two bit polynomial is derived from three one bit polynomial Karatsuba multiplier KM1 in addition some XOR gates are also used. Here a0,a1,b0,b1 are the coefficients of the one degree polynomial and it is given as input to KM2 ; c0,c1,c2 are the output of KM2.In normal classical multipliers requires 4 AND gates and 3 XOR gates. But in Karatsuba multiplier requires only 3 AND gates and 4 XOR gates. Thus the space complexity of the multiplier is reduced.

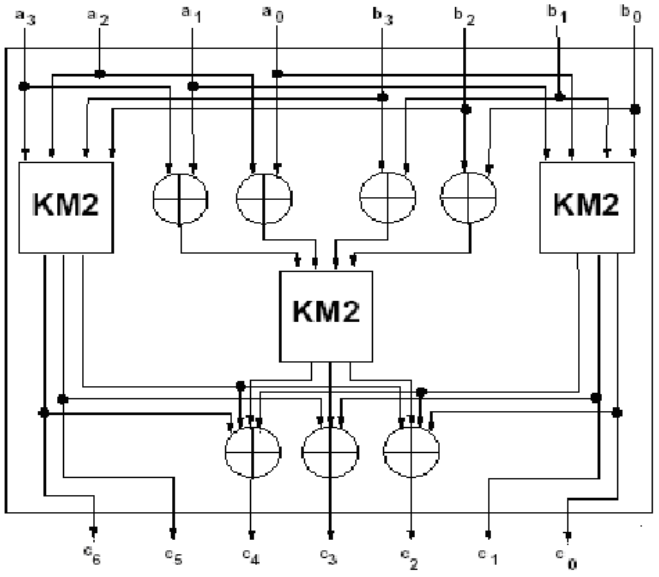

Fig-5: 4- bit polynomial Karatsuba multiplier

Figure 5 shows the 4- bit polynomial Karatsuba multiplier KM4. Here the dot represents the finite field multiplication (AND gate) and the plus represents the addition (XOR gate). The 4-bit polynomial is derived from three 2-bit polynomial Karatsuba multiplier KM2 in addition some XOR gates are also used.

3. RESULTS

In this paper, finite field multipliers are developed and is simulated using Xilinx ISE 8.1i in Verilog. It is synthesized using Xiinx Virtex 4, device Xe3VS500E (package FT250, speed grade -4) and the comparison results are extracted from the synthesis report.

The simulated waveform for the classical multiplier is shown in figure 6. Here the inputs are a= 0101 and b= 1000. The output is c=101000.

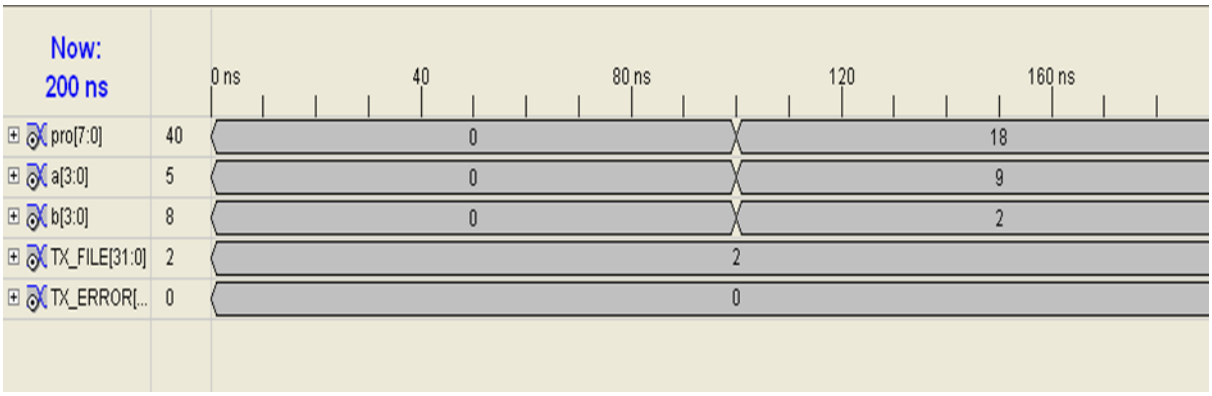

Fig-6: simulated waveform for classical multiplier

The simulated Waveform for Karatsuba Multiplier is shown in figure 7. The inputs given area=1111 and b=0100. The output is c=111100

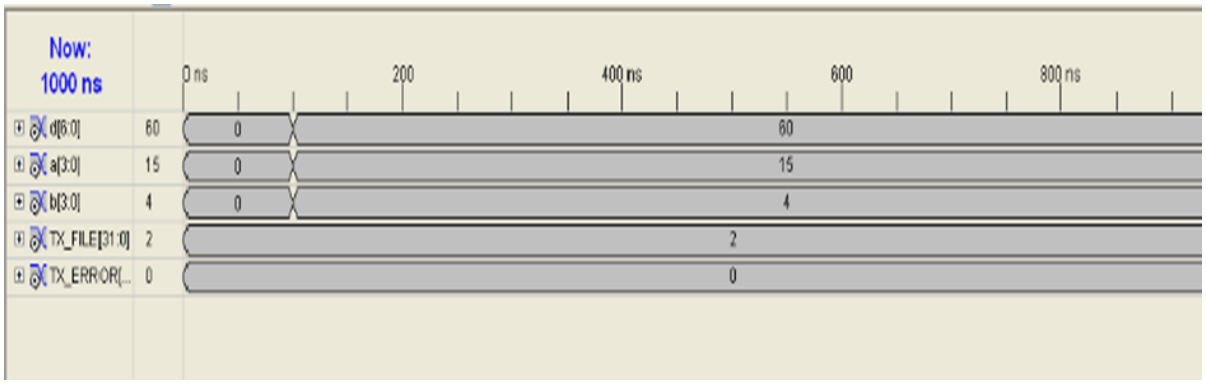

Fig-7: simulated waveform for Karatsuba multiplier

Table-2: Comparison of 4-bit Multipliers

Multiplier	No.of 4-input LUTs	Total Memory Usage (kb)	No.of Slices	Delay (ns)	No. of bonded IOBs
Classical Multiplier	29	118448	17	13.812	16
Karatsuba Multiplier	11	117424	6	7.563	15

Table 2 shows the comparative results of both classical and Karatsuba multipliers. From the table, karatsuba multiplier consumes less space than the classical multiplier.

4. CONCLUSIONS

Finite field multipliers play a very important role in the areas of digital communication especially in the areas of cryptography, error control coding and digital signal processing. In this paper, two multipliers namely classical and Karatsuba multipliers were simulated. The comparison results show that Karatsuba multiplier is more efficient than the other multiplier. Using Karatsuba multiplier we can improve the performance of the process. The Karatsuba algorithm is an optimization technique used for decomposing larger multiplications into multiple smaller multiplications. This feature allows the multiplier to be scaled easily.

REFERENCES

[1]. C.Grabbe,M.Bednara,J.Teich,J.von zur Gathen, and J.Shokrollahi,"FPGA designs of parallel high performance GF(2^{233}) multipliers," in Proc.Int.Symp.Circuits Syst.(ISCAS),May 2003,pp.268-271.

[2]. P.L.Montgomery,"Five,six,and seven-term Karatrsuba – like formulae,"IEEE Trans.Comput.,vol.54,no.3,pp.362-369,Mar.2005.

[3]. C.Paar,"A new architecture for a parallel finite field multiplier with low complexity based on composite fields," IEEE Trans.Comput.,vol.45,no.7,pp.856-861,1996.

[4]. C.Rebeiro and D.Mukhopadhyay,"Power attack resistant efficient FPGA architecture for karatsuba multiplier," in Proc.Int.Conf.VLSI Des.,2008,pp.706-711 N.S.Kim, T.Mudge, and R.Brown, "A 2.3 Gb/s fully integrated and synthesizable AES rinjdael core," in proc. IEEE Custom Integrated Circuits Conf., 2003, pp.193-196.

[5]. A.Reyhani-Masoleh and A.Hasan,"Low complexity bit parallel architecture for polynomial basis multiplication over GF(2^m)," IEEE Trans.Comput.,vol.53,no.8,pp.945-995,Aug.2004.

[6]. F.Rodriguez-Henriquez and C.K.Koc,"On fully parallel karatsuba multipliers for GF(2^m),"in Proceeding(394)Computer Sciences and Technology.Cancun,Mexico:ACTA Press,2003Matthew P.Young June 1, 2006 "Basics of Elliptic curves".

[7]. B.Sunar,"A generalization method for constructing subquadratic complexity GF (2^K) multipliers," IEEE Trans.Comput.,vol.53,no.9,pp.1097-1105,sep.2004.

[8]. J.Von zur Gathen and J.Shokrollahi,"Efficient FPGA-based karatsuba multiplier for polynomials over F_2 ," Sel.Areas Cryptography,vol.LNCS 3897,pp.359-369,2005.

[9]. A.Weimerskirch & C.Paar, " Generalizations of the karatsuba algorithums for efficient implementations,"2006.[Online].Available:http://eprint.iacr.org/2006/224.pdfCharlie Kaufman,Radia Perlman and Mike Speciner "Network Security,"Second Edition, Prentice Hall India.

[10]. G.Zhou & H.Michalik, "Efficient and high-throughput implementations of AES-Gcm on FPGAs" in Proc.Int.Conf.FieldProgram.Technol.(ICFPT),Dec.2007,pp.185-192.